LAMP 精品书廊

悦知文化 Delight Press

Broadview® www.broadview.com.cn 博文视点

Linux 系统架构与目录解析

适用于Linux kernel 2.6以上版本

邱世华 著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

◆ 架构 ◆ 流程 ◆ 管理 ◆

Broadview®
技术凝聚实力·专业创新出版

◆ 架构 ◆ 流程 ◆ 管理 ◆

Linux 系统架构与目录解析

这是一本剖析Linux经典与常用目录及文件的专著，打破了以往只依赖命令语句的惯性思维，以系统目录架构为主体，并设计查询功能，协助读者建立对Linux操作系统的整体认识，而不再仅限于对某些服务或设置的片面了解。

经典特色

- 提供完整的系统启动流程图，为读者说明开机流程中各个目录与文件的重点。
- 以Linux kernel为基础，列出基础且必要的目录结构，可适用于以Linux kernel为核心的各类操作系统。
- 对于修改系统、设置服务或设置X Window等必须熟悉的设置文件，以专章篇幅作详细的介绍。
- 根据各类应用程序在执行时所需载入的函数库文件，以简明易懂的方式说明其规则或标准。
- 完全解析“家目录”的功能，扩展用户登录的流程，以及X Window的操作功能。
- 充分掌握日志文件与暂存信息，引导读者做好“系统管理”的核心工作。
- 以功能分类的方式，将本书中的各项目录及文件制作成索引，方便读者查询与使用。

邱世华 Juergen S.H. Chiu

台湾鸿海精密股份有限公司资深工程师，拥有RHCE（Red Hat Certified Engineer）资格认证，并任教于台湾多所院校。

专长
- Linux系统硬件验证与管理
- Linux Clustering Tech（HPC HA、Loading balance）
- Linux各种服务器的管理
- Red Hat / SuSE Certification tool
- 网络TCP/IP协议
- 刀锋服务器管理

著作 《Linux操作系统之奥秘》《Linux系统架构与目录解析》

悦知文化 Delight Press

图书分类 操作系统

ISBN 978-7-121-08250-4

9 787121 082504 >

定价：45.00元

责任编辑：陈元玉
责任美编：杨小勤

Linux 系统架构与目录解析

邱世华　著

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

这是一本剖析 Linux 常用目录及文件的专著，它打破以往图书偏重介绍命令语句的惯性思维，以系统目录架构为主体，并设计查询功能，以协助读者建立对 Linux 操作系统的整体观念，而不再仅限于对某些服务或设置的片面了解。

本书提供了完整的系统启动流程图，为读者说明各个目录与文件在开机流程中的作用。以 Linux kernel 为基础，列出基础且必要的目录结构，可通用于以 Linux kernel 为核心的各类操作系统。对于修改系统、设定服务，或是设定 X Window 等必须熟悉的配置文件，以专章篇幅做详细的介绍。依各类应用程序在执行时所须加载的函数库文件，以简明易懂的方式说明其规则或标准。完全解析“主目录”的功能，延伸用户登录的流程及 X Windows 的操作功能。充分掌握日志文件与暂存信息，引导读者做好“系统管理”的核心工作。以功能分类的方式，详列本书中的各项目录及文件的索引，方便读者查询与使用。

版权贸易合同登记号　图字：01-2009-0640

图书在版编目（CIP）数据

Linux 系统架构与目录解析 / 邱世华著.—北京：电子工业出版社，2009.4
ISBN 978-7-121-08250-4

I. L… II.邱… III.Linux 操作系统 IV.TP316.89

中国版本图书馆 CIP 数据核字（2009）第 016394 号

责任编辑：陈元玉
印　　刷：北京市天竺颖华印刷厂
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×980　1/16　　印张：17.25　　字数：300 千字
印　　次：2009 年 4 月第 1 次印刷
定　　价：45.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

准备要出版本书的时候，我的心情挣扎了许久，我想读者从书名就可以猜到原因，因为谁会将 Linux 一个目录一个目录逐一打开，并把所有目录及文件都浏览一次，我想应该很少有人会这么做吧！但我还是决定将本书完成，因为——若我今天不写，这样的书何时才会出现？

还记得刚开始学 Linux 时，我一直都非常希望有这样一本工具书可供查询，但可以参考的中文书籍及能够询问的使用者太少，因此，我学习 Linux 的瓶颈也就在这里。虽然比起以前，现在 Linux 的玩家多了许多，但走路、上课、上班时转头看一下周围的人，有几个会用 Linux？毕竟还是少数……

因此，我想就开始写吧！不管出来的结果好与坏，但至少有人在做这样的事情。如果读者看过笔者的第一本著作《Linux 操作系统之奥秘》，相信不难发现，这两本书的内容和市面上 Linux 方面的书籍大不相同，因此，这也是我当初想要出书的动力与目的。改变市面上 Linux 书籍的内容与方向是我的目标，如果市面上每一本书都大同小异，即使出了一千本 Linux 书籍，读者可参考的有实际价值的也可能不会超过三本，因为书的结构、内容和包装都十分相似，这样对读者的 Linux 学习之路将没有多少助益。

虽然在写作本书期间，我找了非常非常多的数据，但我绝对不敢说本书把所有 Linux 的目录及文件都纳入了其中，毕竟一人之力有限， Linux 的版本又如此之多，所以我只能将我所知的及可以查到的所有信息列在其中，以呈现在读者面前。如之前《Linux 操作系统之奥秘》一样，本书读起来可能会比较枯燥（希望读者以工具书的方式查阅），但我想接下来要撰写的书籍，应该会写一些稍微轻松的主题，至于有哪些？等出来后就知道啰！

不过，还是要提醒读者，不要指望靠一本书，就可以掌握你正在使用中的 Linux，因为 Linux 的 kernel 版本、发行（Distribution）版本都有可能影响到里面的结构与操作，如在写本书时，我还是以较常用的 Fedora 为主，至于是否适用 SuSE、Ubuntu、Debian 或其他的版本，我想不会百分之百适用，但绝大部分都是通用的，如/proc、/sys、/dev 等，因为底层都是 Linux kernel，其他不一样的，须看开发厂商是按哪一种标准、哪一种方式或哪一种功能去分配该版本的目录或软件的，但都必须参照 Linux 的基本精神去开发。在此要告诉读者的是，虽然写本书

笔者只用了一个版本，但读者是可以变成好几个版本的。Linux 版本虽然很多，但所需要的书并不多，你只要找到几本好书，了解 Linux 的基本精髓，用一样的方式去读每一个 Linux，这样才是学会 Linux 的最快方式。然而，每一本书有每一本书所适合的版本，你永远不可能知道这本书何时会被淘汰掉，只有懂得最基本的概念，适应手边的 Linux，才可以掌握每一版 Linux 所需的要素。

本书耗时快一年才完成，比起上一本书，时间长了许多，一方面是最近真的很忙，但最主要的原因，是因为第二个小孩也出生了，相信生过小孩的，就知道笔者目前的辛苦了！在家完全无法做事，可能也因为我是一个不太“敢”凶小孩的爸爸。因为刚要开始写就遇到了第二个宝贝的“关爱”，进度就“微微”落后，也就受到编辑“轻轻”的叮咛。

然而，我也希望在小孩长大后，回过头看到“当初”爸爸在这段时间所写出来的东西（只能希望他们会用 Linux 啰），是有点与众不同的，具有自己的想法，这样也可以借以告诉他们，如果要想把一件事情做好，就必须要有自己的想法、目标，不管别人是如何做到的，只要管好自己做得好不好就行。亦如我在规划写本书时，从没想过有没有可以参考的书籍，或者是有没有人写过，如果去参考，写出来的东西，就会有别人的看法、别人的思想或影子在里面，就不会是完全属于自己的东西。因此，要做的事情贡献大不大，是不是自己要的，这是唯一重要的。只要做的事是对的，就一定会有人欣赏。

尽管这是笔者撰写的第二本书，但我还是想献给每天辛苦在家带小孩带到快疯掉的老婆（或称大美女），希望她看到本书后能带给她多一点点的安心（可以帮小孩多买几本书了）；还有我好爱好爱的小 baby，当本书出来的时候，不知道她会不会让我亲亲她的脸，而不用先向妈咪报告；当然也要献给已经快被我狂亲（有点像在吃了）一年的小壮丁（号称米其林宝宝），虽然第一本书你来不及以“本尊”参与，但第二本书也千万不要拿去当点心吃掉，我手上的书可能不够你吃的，等你一岁再买蛋糕给你吃啰！

在此也要谢谢所有我第一本书的读者，因为有你们的肯定，我才会有力量出版第二本书（说实话是出版社才敢让我出），也希望你们继续给我意见及支持，如果对本书有任何好的想法，同样希望可以告诉我或出版社，因为 Linux 的东西原本就是要靠众人的力量，才可以变得更好，不是吗！

邱世华
Juergen.chiu@gmail.com
2008 年 8 月

导读

笔者从一开始写本书，就定义为一本工具书。也就是说，希望读者不要一次从头看到尾，因为书中的目录及文件实在太多太多。若以这样的方式看完本书，可能就好像拿一本英文字典从头看到尾的感觉，只是多了一些图片和说明。笔者只能在本书中，用一些功能、文件系统的方式将大量的目录及文件做一些基本的分类，美中不足的是无法让整本书有连贯的关联，或许这是有待加强的地方。

笔者希望读者将本书放在手边，在使用 Linux 过程中，遇到任何有疑问的目录或文件，可以利用该目录文件，或是上一层的目录文件，做实时的查阅，这样能够增进学习的速度，减少许多需要问别人的时间，或是等待别人回答的时间。另外，本书也可以帮助读者在发生问题时，通过书籍内容的介绍，让读者知道有哪些方式，或者应该如何对产生问题的目录文件做相对应的动作，不然往往在问题发生时，会不知道该目录或文件是不是可以做一些新建、删除、修改或忽略等动作，这也是本书希望可以带给读者的应用。

在此先将整个 Linux 的目录文件分为以下几个章节。

第 1 章：Linux 目录的基本概念

在这一章中，试着要告诉读者的是，在用户看到所有目录之前，Linux 到底是按照怎样的标准或程序，将目录挂载进系统的，毕竟凡事都有一个源头及根基，知道这些事情，才可以帮助读者更深刻了解目录或文件所代表的意义。

第 2 章：不同启动模式的目录

操作系统或系统的应用方式不同，启动的模式可能都会不同，尤其现在 Linux 被应用的范围很广，已经没有人敢说某一个操作系统一定是通过硬盘启动，有可能是用一个硬盘、U 盘或网络就可以进入 Linux，而其中的差异就会牵涉到使用不同的 Linux 目录。

第 3 章：Kernel Space 与 User Space 的桥梁——虚拟文件系统

Linux 下面有许多目录或文件，不一定是真的存在于一般的文件系统下（像 ext3），很多是通过特殊的渠道让用户可以看到，但并不是真正的目录或文件，可能只是一个功能的开关，而这些都属于虚拟的文件系统，在这一章会有详细的介绍。

第 4 章：应用程序目录

每一个操作系统都会有许许多多的应用软件，但这些应用软件所需要的执行文件或所须载入的函数库都放在哪里？本章会将大部分应用软件所遵循的规则或标准整理出来供读者参考。

第 5 章：用户的主目录

当用户一登录到 Linux 操作系统时，就会拥有一个有完全使用权限的目录，我们称为“主目录”。在这一目录下，用户可以完全的操作权限，但如果不知道主目录中有哪些目录或文件可以改变一般用户的登录流程或 X Window 的操作，就会浪费主目录的权限了。

第 6 章：系统配置目录

很恐怖的一章，因为文件太多了，光是这一章就要占用非常大的篇幅，当然，在这下面的主要目录或文件，都是一些系统及服务相关的配置文件，如果要熟悉如何修改系统、配置服务或是 X Window 的配置，本章的内容是一定要看的。

第 7 章：日志文件与媒体挂载目录

要管理好一台系统，有一点很重要，就是要完全掌握每一个系统所发生的信息都输出到哪了，本章就是将大部分读者会使用到的信息或临时信息展现出来，让读者在以后管理系统时，不会不知道该错误信息或临时配置文件存放的位置。

为了在最短的时间内找到某一些特定的目录或文件，笔者用了两种方式让读者方便索引。

流程图

在每一个章节的最前面会以系统完整的流程图（如下图所示）展现，表现出该章节中的目录可能会影响到系统的哪一个阶段，虽然一定无法完全涵盖所有范围，或是百分之百地正确（启动所需的文件相当多），但至少可以让读者在发生问题时，可在最短的时间内判断出问题所在的目录或文件，进而在本书中查询该目录文件的内容，以通过本书的介绍解决该系统的问题。

此外，每一章的重点与说明流程都不相同，所以在最开始会标示出本章的学习重点，让你

从示意图中了解目前在哪一个阶段。

目录索引

本书制作了让读者可通过章节、功能分类的索引，让读者在阅读或查询时，可以在最短的时间内找到想要得到的资料，这可是花了笔者及编辑很多时间才整理出来的。

另外，在悦知的网站上，也已经将 Linux 的目录结构标准——FHS 的官方文件，放在本书的网页中，虽然 FHS 标准在第 1 章已经有大概的描述，但需要这份文件的读者，还是可以直接到 FHS 官方网站或博文视点官方博客网站上下载。

博文视点官方博客网址

http://blog.csdn.net/bvbook

FHS 官方网站

http://www.pathname.com/fhs/

联系博文视点

您可以通过如下方式与本书的出版方取得联系。

读者信箱：*reader@broadview.com.cn*

投稿信箱：*bvtougao@gmail.com*

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特 1 号 湖北信息产业科技大厦 1402 室

邮政编码：430074

电　　话：027-87690813

传　　真：027-87690595

若您希望参加博文视点的有奖读者调查，或对写作和翻译感兴趣，欢迎您访问：*http://bv.csdn.net*

关于本书的勘误、资源下载及博文视点的最新书讯，欢迎您访问博文视点官方博客：*http://blog.csdn.net/bvbook*

目　录

索　引

第 3 章：Kernel Space 与 User Space 的桥梁——虚拟文件系统

/dev/

/proc/

/proc/

/proc/

/proc/

/sys/

第 4 章：应用程序目录

/bin/

/usr/

/tmp/

第 5 章：用户的主目录

/home/

/home/

第 6 章：系统配置目录

/etc/

/etc/

/etc/

/etc/

/etc/

/etc/

/srv/

第 7 章：日志文件与媒体挂载目录

/var/

/var/

第 1 章　Linux 目录的基本概念

本章学习重点

- Linux 被定义的基本目录
- 什么是根目录
- 根目录是如何产生的
- 根目录有何意义
- 本书所提到的目录有哪些

Linux 的目录是很多用户想了解但一直无法进入的门槛，这句话的意思是，或许很多人试着想要知道，到底 Linux 有哪些目录、哪些文件、哪些是值得去刨根问底的。但大多数人，都会因为庞大的目录及文件数量而却步，我想这也是为何到现在为止，还没有一本书愿意介绍所有目录架构的原因。

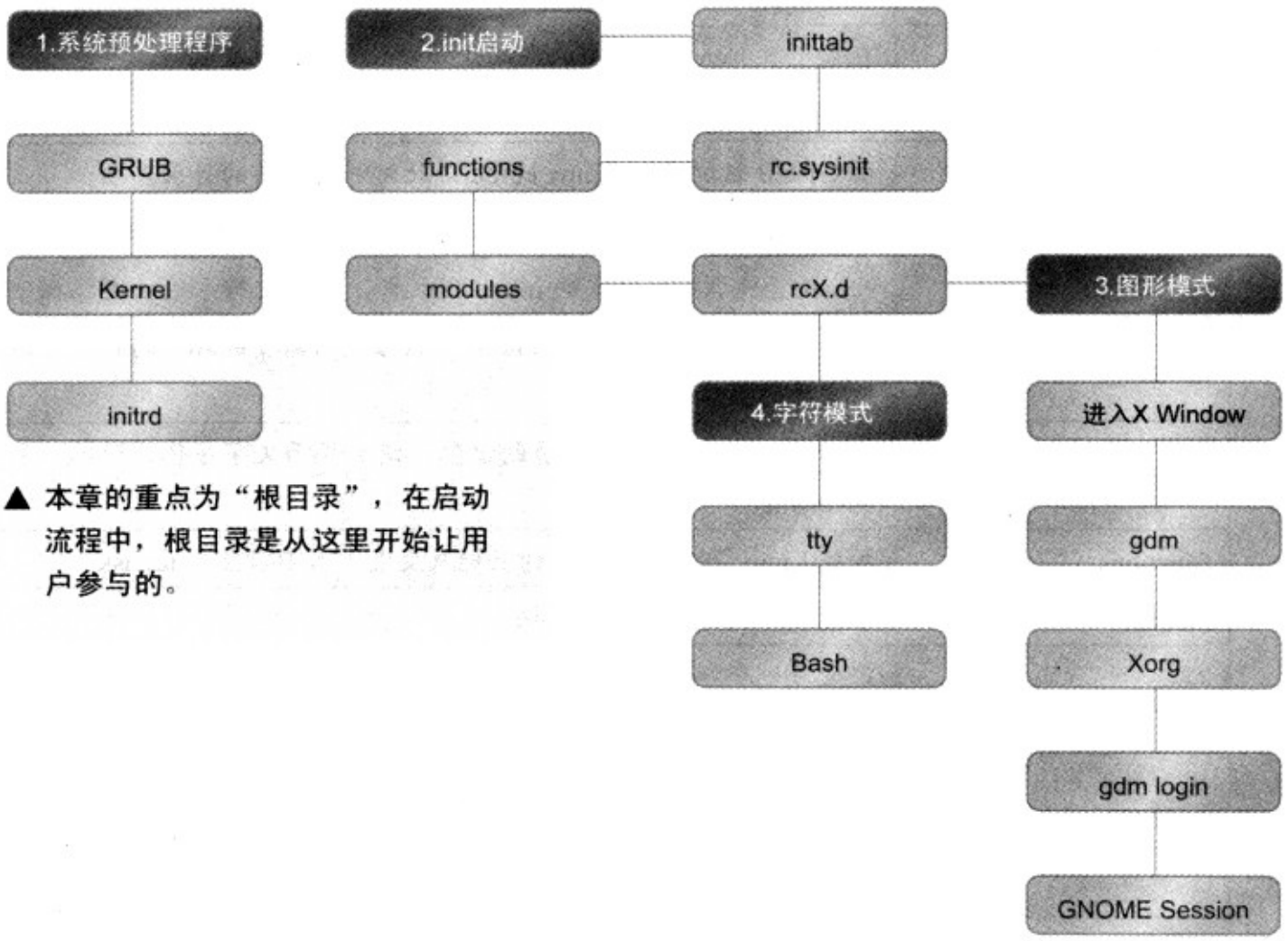
不过，当各位读者在看这样一本书时，千万不要认为所有的 Linux 版本（如 Red hat、SuSE）都会是一致的，本书只能作为参考，主要原因是不同的操作系统厂商，所安装的组件、遵循的标准、版本的差异等都会造成整个 Linux 下目录或文件的不同（像有些服务程序放在/etc/rc.d 下，某些则是在/etc/init.d 下的差异），但是，只要熟悉其中一种的目录结构，最起码在很短时间内就可熟悉另一种操作系统。

不管如何，目录及文件在整个 Linux 启动运行过程中，不断地被使用，只是每一个阶段使用到的目录会不同，所以无法完全对照出整个系统运行流程和目录的关系，但笔者尽量将其归纳为一个整体性的图（如下图所示），以及对照到此书章节的表格，希望对读者在阅读本书时会有所帮助。

这张图会在本书中每章的最前面来提醒读者所阅读的部分和系统流程中，哪些部分是相关的，让读者更了解 Linux 目录的整体关系。

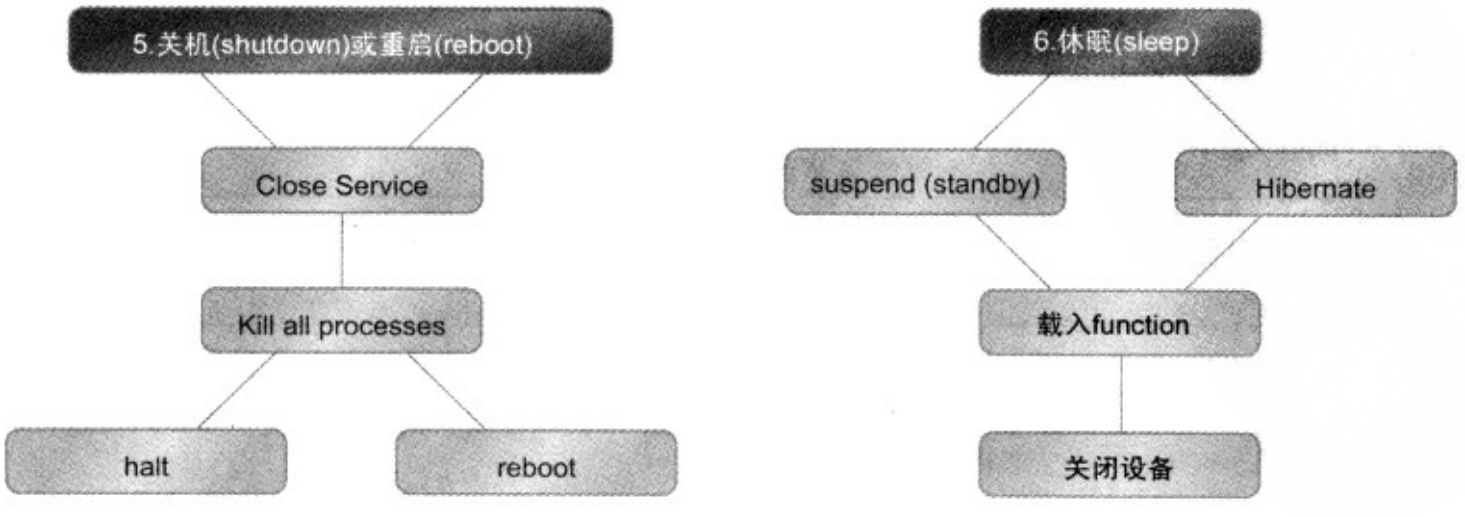
系统流程与章节内容对照图

- 启动流程



▲ 本章的重点为“根目录”，在启动流程中，根目录是从这里开始让用户参与的。

- 关机流程



启动流程中所使用的重点目录对照表如表 1-1 所示。

表 1-1：启动流程中所使用的重点目录对照表

编号	登录阶段	图中说明
1	系统预处理程序	在未进入 Linux 系统前的一些基本操作
	GRUB	引导管理程序，提供用户开机画面及操作系统菜单
	Kernel	由 GRUB 协助载入 Linux kernel，此时并无任何的目录产生
	initrd	由 GRUB 协助一并载入 initrd 文件，让 kernel 支持更多硬件
2	init 启动	kernel 直接调用系统中的 init 执行文件，开始进行系统的运作程序
	inittab	这是 init 的脚本文件，让 init 知道该系统所需要的环境与每一阶段所需要的步骤
	rc.sysinit	开始进行 Linux 系统环境的配置，诸如界面文字字体、时间、系统记录等
	functions	加载 Linux 系统一些内置的功能及变量，如 PATH 或 umask
	modules	这时会加载一些默认的模块
	rcX.d	定义每一个 runlevel 系统阶段所需要的服务有哪些，大部分不是文字模式就是图形模式，其他都是较为特殊的应用
3	图形模式	进入图形模式界面
	进入 X Window	inittab 中会记载要以哪一种模式进入 X Window，主要是注明加载 X Window 的管理程序是什么
	gdm	一般 Redhat 系统都是以 gdm 为主要的管理软件，因此，在进入 X Window 前，须先启动 gdm 管理程序
	Xorg	gdm 会先协助用户将 X Window 的基本环境启动，而 Xorg 就是 X Window 的主要环境接口，有了 Xorg 才有办法启动其他的应用程序
	gdm login	这是经过 gdm 所特有的登录界面，可以帮用户以不同的账号使用不同的 X Window 配置值
	GNOME Session	在这阶段用户拥有完全的 X Window 功能，包括所有的软件画面
4	文字模式	进入文字模式接口
	tty	开始出现控制台画面，并提供让用户可以进入系统的登录画面
	bash	登录后，大部分 Linux 所提供给用户的操作环境，都是用 bash 作为底层环境的

续表

编号	登录阶段	图中说明
5	关机或重启	不论关机或重开，其实都是做差不多的事情，都须要将所有的程序清除
	Close Services	关机时须先将所有服务停止
	Kill all processes	停机前会将所有在内存中的程序先清空
	halt or reboot	关机或重启的最大差别，就在于最后执行哪一道指令（halt 或 reboot）
6	休眠	不关机却可以省电的方式，就是进入休眠模式
	Suspend/Hibernate	休眠最多使用的模式，就是 Suspend 和 Hibernate，最大的差别，就在于要将系统现阶段的状况记录在硬盘（Suspend）或是内存（Hibernate）中
	加载 function	将休眠所需要的一些功能指令加载到系统中，以进行实际的休眠动作
	关闭设备	将所有的存储设备停止动作

因为 Linux 是开放源代码的系统，所以每一家 Linux 厂商（如 Redhat、SuSE、Ubuntu 等）都有自己的考虑，会加入和删除一些基本选项或特殊功能，也因为这样，每一家厂商所推出的 Linux 发行版（Linux Distribution），其目录的文件数目一定会有相应的差距，甚至是同样组件中的某个文件，文件名称也会随 Linux 版本而有所差异，因为控制权其实是握在厂商的手上。

当然也会造成用户的麻烦，因为明明都是 Linux 系统，但是要在另一种平常没在使用的 Linux 版本中，寻找某一个经常使用的配置文件或指令，会发现竟然找不到，花费了很多时间才发现，可能只是一两个字符的差异。或者是 A 版本默认有该文件，而 B 版本默认没有该文件，其实都是可以使用的，但在不知情的情况下，也会造成用户的困扰，因此，有人开始规划一个让所有 Linux 版本都可遵循的目录规则。

FHS 组织（Filesystem Hierarchy Standard Group）就是在这样的背景下应运而生的，该组织定义了一份 FHS 的公开标准，让所有 Linux 厂商参考，目的就是让大家都可以在同样的条件下发展所有的应用程序，将该有的目录或文件都可以因为使用同一份标准规范，而让用户有一个统一的方式。

当然，无论是哪一个目录或文件，Linux 目录中最基本的就是【/】，所有使用 Linux 的用户，都会知道【/】根目录的重要性，但这个目录是何时建立与为何而产生的，就比较少有人去刨根问底地看个清楚了，这也是本章的学习目的之一，同时可以和读者分享根目录的意义所在。

1.1 Linux 目录的定义

Linux 并没有硬性规定该有哪些目录或文件，大部分数据都是由 Linux 厂商自行处理的，只是因为大家所使用的都是 Linux 的 kernel，所以基本上都会有相似的文件系统架构，但也因为如此，Linux 的用户在转换上有时会产生一些瓶颈。

如在 Fedora 7 下的与 SuSE 10 下的网卡配置文件，文件名称的命名方式就有很大差异（如图 1-1 和图 1-2 所示），Fedora 7 的命名规则为【ifcfg-适配器名称】，但 SuSE 下则为【ifcfg-eth-id-网卡的 MAC Address】。

虽然这只是因为各家的网络配置方式不同，而采用的文件命名原则不同，但从图 1-1 中一样可以发现，两个系统连启动网络接口的工具程序也都放在不同的位置，这对经常使用固定某一版本的 Linux 用户来说，刚开始是会有一些麻烦的。

```
[root@LinuxTree network-scripts]# pwd
/etc/sysconfig/network-scripts
[root@LinuxTree network-scripts]# ls
ifcfg-eth0    ifdown-isdn    ifup-aliases  ifup-plip     ifup-wireless
ifcfg-lo      ifdown-post    ifup-bnep     ifup-plusb    init.ipv6-global
ifdown        ifdown-ppp     ifup-eth      ifup-post     net.hotplug
ifdown-bnep   ifdown-routes  ifup-ippp     ifup-ppp      network-functions
ifdown-eth    ifdown-sit     ifup-ipsec    ifup-routes   network-functions-ipv6
ifdown-ippp   ifdown-sl      ifup-ipv6     ifup-sit
ifdown-ipsec  ifdown-tunnel  ifup-ipx      ifup-sl
ifdown-ipv6   ifup           ifup-isdn     ifup-tunnel
[root@LinuxTree network-scripts]#
```

Fedora 7下的网卡设定档配置文件

这里所看到的很多网络设置工具（如ifup或ifdown开头的文件），在SuSE中都放在"scripts"的子目录下

图 1-1：Fedora 7 下网卡配置文件的位置与文件名

```
linux:/etc/sysconfig/network # pwd
/etc/sysconfig/network
linux:/etc/sysconfig/network # ls
.   config  if-down.d  ifcfg-eth-id-00:0c:29:ed:ec:67  ifcfg.template  providers  wireless
..  dhcp    if-up.d    ifcfg-lo                        ifroute-lo      scripts
linux:/etc/sysconfig/network #
```

图 1-2：SuSE 10 下网卡配置文件的位置与文件名

FHS 组织为了解决这样的难题，因此制订出一份 FHS 标准，希望所有 Linux 厂商可以比照其所定义出的目录结构（基本上只制订大方向，毕竟不可能定义到太多层的目录结构）来开发应用程序，这样一来，对厂商或用户都有好处，一方面厂商可以加快开发的速度；另一方面用户可以缩短适应另一种 Linux 的时间。

笔者在这里简单地列出 FHS 制订的标准（表 1-2 顺便和 Fedora 7 目前的目录结构做简单的

比较），目前 FHS 已经出到 2.3 版（于 2004 年 1 月 29 日发行）以供读者快速参考，但详细的内容还是请读者直接参考 FHS 官方网站上的标准文件。网址如下：

http://www.pathname.com/fhs/

表 1-2：FHS 所定义的目录结构

	FHS 的定义	Fedora 7 默认是否有该目录存在
Linux 的核心目录结构		
/bin	存放所有用户都可以使用的“必要”命令	YES
/boot	存放开机启动加载程序的核心文件	YES
/dev	设备文件目录	YES
/etc	主机或系统配置文件目录	YES
/etc/opt	/opt 下应用程序的配置文件目录	YES
/etc/X11	X Window 的配置文件目录	YES
/etc/sgml	SGML 的配置文件目录	YES
/etc/xml	XML 的配置文件目录	YES
/home	存放用户的用户目录	YES
/lib	需要共享的函数库与 kernel 模块	YES
/lib<qual>	当一个系统中有不同版本的/lib 时所使用	NO
/media	移动存储设备的挂载点	YES
/mnt	临时挂载用的挂载点	YES
/opt	额外所安装的应用程序目录	YES
/root	管理员的主目录	YES
/sbin	系统的“必要”指令文件存放区	YES
/srv	系统服务的文件存放区	YES
/tmp	临时文件暂存区	YES
/usr 下的目录结构		
/usr/X11R6	X Window 系统的 Version 11 Release 6 版本主目录	YES
/usr/bin	存放所有用户都可使用的“非必要”指令	YES
/usr/include	标准的表头文件存放区	YES
/usr/lib	存放程序所使用的函数库	YES
/usr/lib<qual>	当一个系统中有不同版本的/usr/lib 时所使用	NO
/usr/local/share	和/usr/share 存放相同的数据	YES

续表

	FHS 的定义	Fedora 7 默认是否有该目录存在
/usr/sbin	系统的“非必要”指令文件存放区	YES
/usr/share	软件架构保持和系统独立时使用的目录	YES
/usr/share/dict	字典文件目录	YES
/usr/share/man	使用手册目录	YES
/usr/share/misc	和系统独立的杂项文件目录	YES
/usr/share/sgml	SGML 文件目录	YES
/usr/share/xml	XML 文件目录	YES
/usr/src	源文件目录	YES
/var 下的目录结构		
/var/account	储存程序审核的记录	YES
/var/cache	应用程序的缓存文件目录	YES
/var/crash	宕机时导出（dump）信息的目录	NO
/var/games	游戏软件的文件	YES
/var/lib	各种软件的状态信息	YES
/var/lib/<editor>	储存编辑器的备份文件或状态	NO
/var/lib/hwclock	RTC 的状态目录	NO
/var/lib/misc	储存在/var/lib 中除状态信息外的杂项数据文件	YES
/var/lock	lock 文件储存区	YES
/var/log	记录文件存放目录	YES
/var/mail	用户的邮件信箱	YES
/var/opt	在/opt 中的应用程序所使用的 var 目录	YES
/var/run	正在执行中的运行时文件	YES
/var/spool	应用程序的队列文件目录	YES
/var/spool/cron	cron 和 at 的计划任务文件	YES
/var/spool/lpd	打印机的队列文件存放目录	YES
/var/spool/rwho	存放 Rwhod 的文件	NO
/var/tmp	系统在重启之间所需要的暂存区	YES
/var/yp	NIS 数据库的文件	YES

除了 FHS 所定义的目录之外，每一家 Linux 厂商都有额外自行建立的目录，供自行开发的应用程序使用，所以真正的目录结构还是要以用户实际使用的 Linux 版本为依据，只是大部分目录不会脱离 FHS 所定义的标准。本书大多以 Fedora 7 为例，这并不代表任何特殊含义，只是希望通过一个操作系统的版本（因为笔者比较常使用的操作系统是 Fedora 7）去阐述每一个目录的意义与所要提供的功能，毕竟各 Linux 版本的目录是大同小异的。

当然，笔者也并非万能，只能尽其所能地让读者对每一个目录有所了解，至于每一个目录涉及的深度，有可能会因为每一个目录的功能多少、文件数目、对系统的意义大小或笔者本身能力而有所差异。

1.2 根目录的建立

大家一般都会知道根目录的产生方式，就是系统使用 mount 指令，将系统所在的分区挂载到【/】目录中，这样便完成了所谓的根目录。但你是否想过，虽然看起来合理却有点诡异，因为根目录既然是 Linux 的“根”，那没有根，哪来的 mount 指令？系统怎么可以使用呢？这正是要在这一节解释的部分。

讲到如何产生根目录，必须先知道根目录产生之前的一些基本系统运行动作（如图 1-3 所示），在开机管理程序启动操作系统，在加载 kernel 之后（也就是当用户在开机画面选择某系统选项按【Enter】后），kernel 会自行在内存中建立一块叫做 rootfs 的区域供本身使用，而里面的功能都是 kernel 本身所提供的，这也就是编译 kernel 时所赋予的能力，不过大部分 kernel 的能力都是在安装完操作系统后就已经定义好了的，除非是自行重新将 kernel 编译过。

而这一段 kernel 执行的过程，并不是产生根目录【/】的阶段，也就是说，在 kernel 启动阶段，并没有使用到根目录，而根目录产生出来的时间点，是在 kernel 加载完成后，下一个 initrd（Initial ramdisk）加载模块期间。

这其实是因为在尚未加载 initrd 之前，如果操作系统是在网络或 SCSI 接口上，必须要等到相关模块加载后才可以使用（除非是手动将该模块嵌入到 kernel 中），在这种情况下，如果 kernel 不支持该存储设备或功能（像 SAN），如何能辨认及使用正确的根目录？所以，必须等到 initrd 加载正确的模块，并且正确地辨认出存储设备的硬件之后，才能将系统分割区准确地挂载到根目录上，产生出一般使用的【/】根目录。

从这一段启动的信息（如图 1-4 所示）可看出，基本上经历了以下 3 个步骤，根目录被产生出来，不过不同厂商所推出的 Linux，有可能会有不一样的动作，但概念是差不多的，图 1-4 所圈选的部分，是用户可以通过启动信息所看到的建立基本目录的三大步骤。

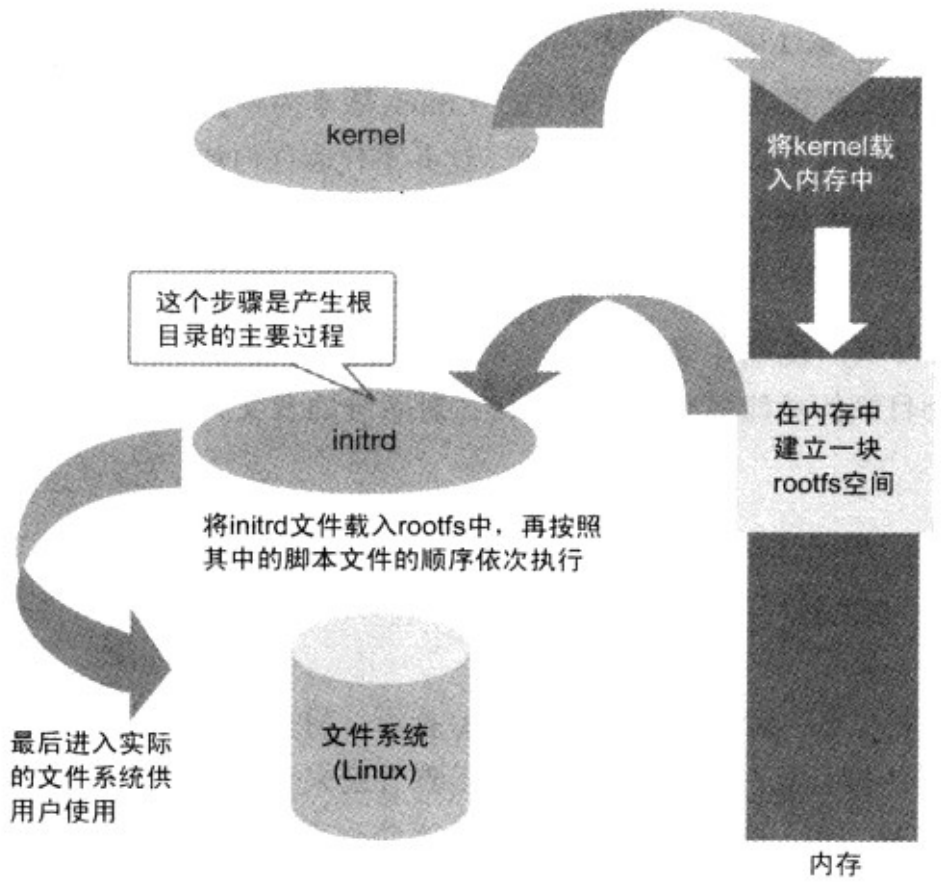


图 1-3：进入系统前的运作流程

Step 1 建立根目录所需的设备文件。

Step 2 先将该设备文件所指的位置（分割区）挂载到【/sysroot】目录下。

Step 3 切换目录到根目录下。

```
Loading dm-mod.ko module
device-mapper: ioctl: 4.11.0-ioctl (2006-10-12) initialised: dm-devel@redhat.com
Loading dm-mirror.ko module
Loading dm-zero.ko module
Loading dm-snapshot.ko module
Making device-mapper control node
Scanning logical volumes
  Reading all physical volumes.  This may take a while...
  Found volume group "VolGroup00" using metadata type lvm2
Activating logical volumes
  2 logical volume(s) in volume group "VolGroup00" now active
Trying to resume from /dev/VolGroup00/LogVol01
No suspend signature on swap, not resuming.
Creating root device.
Mounting root filesystem.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: mounted filesystem with ordered data mode.
Setting up other filesystems.
Setting up new root fs
no fstab.sys, mounting internal defaults
Switching to new root and running init.
unmounting old /dev
unmounting old /proc
unmounting old /sys
```

图 1-4：产生根目录的时间点

完成这一连串的动作后，重要的【/】根目录就这样被 initrd 产生完毕，接着就是等待根目录下/sbin/init 执行文件启动，由 init 文件将其下所负责的软件或服务依序执行，就可以将整个 Linux 所需的环境建立完成。

1.3 根目录的意义

根目录是做什么用的？个人认为它是一个目录中最基本的地址代号，如果把一个网站“比喻”为 Linux，那网站主要的域名（Domain Name）就等于是 Linux 的根目录，而网站的子目录则代表根目录下的所有目录名称。

譬如说，以下网址是代表【/】根目录（其实对网站来说，真的是该“网站”的根目录）：

http://juergenchiu.blogspot.com/

那么以下网址，就很像是 Linux 中【/search/label/cluster】目录，这比喻不知道各位是不是可以接受？但我想这是对根目录概念的理解。因为使用根目录就像在浏览网页一样，应该要能随时切换到不同的网站去浏览。

http://juergenchiu.blogspot.com/search/label/cluster

很多时候可以利用这样的概念来完成一些原本无法做的事情。例如，如果同时有两个不同的操作系统存在于硬盘中（像多重引导操作系统，但可以跨硬盘，像 SuSE 和 Fedora 就不一定要在同一个硬盘中），此时，若用户先开机进入 Fedora，但临时有一件事情要到 SuSE 下面操作（譬如，要通过 yast 指令做某些配置，这就很难在 Fedora 下，利用本身的工具程序去完成其他分割区或硬盘中 SuSE 的数据），这时可以先将 SuSE 所存在的分割区挂载上来，再使用【chroot】指令，将根目录切换到该分割区，就可以使用 SuSE 上面的资源，这样是不是很像在浏览网页呀！

步骤描述如下，请同时参考图 1-5 所示的画面。

Step 1 将另外一个 SuSE 操作系统的硬盘（或现存的 SuSE 操作系统分割区）加到 Fedora 7 之下（用别的版本也可以，但只有 Fedora 7 是默认就支持 SuSE 的 ReiserFS 的文件系统格式）。

Step 2 先直接开机进入 Fedora 7 操作系统。

Step 3 此时可以将 SuSE 所存在的硬盘或分割区挂载到 Fedora 7 的【/mnt】目录。

```
[root@localhost ~]# uname -r ; cat /etc/issue
2.6.21-1.3194.fc7
Fedora release 7 (Moonshine)
Kernel \r on an \m

[root@localhost ~]# cat /proc/partitions
major minor  #blocks  name

   8     0   10485760 sda
   8     1     104391 sda1
   8     2   10377990 sda2
   8    16   10485760 sdb
   8    17     514048 sdb1
   8    18    9968332 sdb2
 253     0    9273344 dm-0
 253     1    1048576 dm-1
[root@localhost ~]# mount /dev/sdb2 /mnt/
ReiserFS: sdb2: found reiserfs format "3.6" with standard journal
ReiserFS: sdb2: using ordered data mode
ReiserFS: sdb2: journal params: device sdb2, size 8192, journal first block 18,
max trans len 1024, max batch 900, max commit age 30, max trans age 30
ReiserFS: sdb2: checking transaction log (sdb2)
ReiserFS: sdb2: Using r5 hash to sort names
[root@localhost ~]# _
```

目前在Fedora 7的环境下

sdb是含有SuSE操作系统的硬盘

将SuSE系统的分区挂载到Fedora 下

图 1-5：将 SuSE 的硬盘加到 Fedora 7 的系统中

在额外的操作系统（SuSE）加进来之后，就可以试着在 Fedora 7 的环境下，直接转变为 SuSE 的环境，这是一件非常简单的事！

Step 4 刚刚已经将 SuSE 的操作系统挂载到/mnt 目录下，其实 SuSE 操作系统的所有文件在这个动作后，就已经存在 Fedora 之下（当然也要视 SuSE 安装时的目录规划而定，这边以简单的方式说明）。/mnt 这个目录是可以自己决定的，但要记得不可以和原本已经在使用中的目录冲突，虽然不太可能会有无法挂载的情况发生，但如果将原本使用的目录取代掉，有可能造成系统很多的功能无法使用（如指令）。

Step 5 使用【chroot】指令，将目前的根目录切换到【/mnt】下，也就是 SuSE 的根目录下（见图 1-6）。

```
[root@localhost ~]# chroot /mnt/
localhost:/# cat /etc/issue
Welcome to SUSE LINUX 10.0 (i586) - Kernel \r (\l).

localhost:/# _
```

此时已经将环境切换到SuSE操作系统下

图 1-6：将根目录切换到 SuSE 下

Step 6 这时可以执行【yast】指令（见图 1-7）。

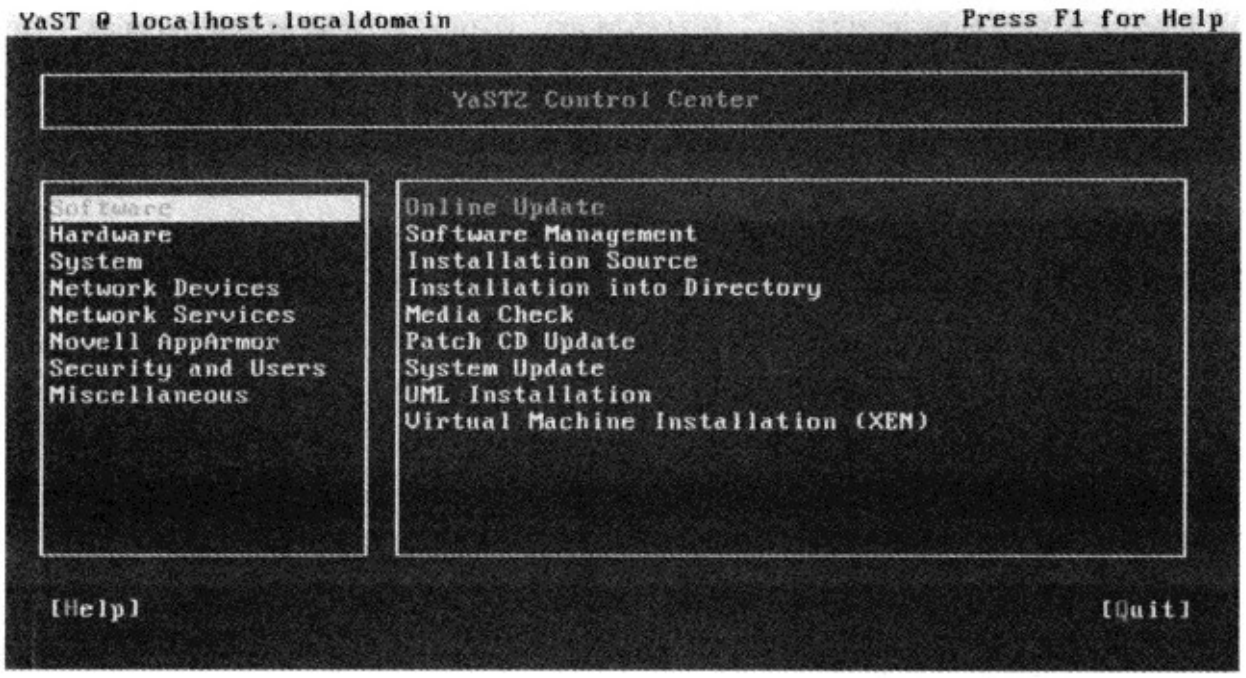


图 1-7：在 Fedora 的操作系统下转变到 SuSE 的软件

Step 7 接下来，可以做一些“基本”的 SuSE 配置或软件的执行，但仅限于 SuSE 的硬盘或目录下。为何只能做到“基本”的动作，因为这时系统还是在 Fedora 的环境下，这代表着，虽然看到的目录、软件都已经变为 SuSE 的样子，但底层的 kernel 还是原本的 Fedora，所以功能较少。图 1-8 简单说明了使用上的限制。

```
localhost:/# uname -r
2.6.21-1.3194.fc7 (A)
localhost:/# ls /lib/modules/
. .. 2.6.13-15-default 2.6.13-15-xen precompiled scripts (B)
localhost:/# modprobe -ligrep bonding
FATAL: Could not load /lib/modules/2.6.21-1.3194.fc7/modules.dep: No such file o
r directory
localhost:/# _ (C)
```

图 1-8：转换根目录后的限制范例

Ⓐ 目前 Linux 操作系统真正使用的是 kernel 版本。

Ⓑ 用户根目录下的模块目录清单，很明显，该模块目录的版本和所使用中的 kernel 版本不符。

Ⓒ 当用户要查询、加载模块时，会因为两者不符，造成错误的信息，因为 kernel 为 2.6.21 版本，但所找到的模块版本目录却是 2.6.13 版本。

1.4 根目录中的目录清单

其实，Linux 中的目录及文件数目多到数不出来，本书也不可能将所有的目录及文件列出，

否则光是列出所有的目录及文件就可以达到一本书的厚度。所以之后章节所介绍的目录（绝大部分会以目录为主，文件视其对系统的影响程度而定，毕竟目录都写了，放着重要文件不管有点浪费），大致会介绍到根目录下的第二层（当然是系统完整安装后的目录），此为本书的核心目标。

光是第二层的目标目录（还未包括文件，加上文件有上千个），就已经有几百个等着我们去研究（如图 1-9 所示，此为刚安装完成时的数目，不同版本或不同的安装方式会造成数目的不同），所以整本书还是会根据目录的重要性来介绍。内容中不会全部都刚好是第二层，有些可能会不到第二层或更深入一点，视目录的重要性而定。

```
[root@localhost ~]# find / -maxdepth 2 -type d | wc -l
327
[root@localhost ~]#
```

图 1-9：本书大致介绍的目录清单取得方式

当然，若读者在看完本书后，觉得有些想知道但没有被列出的目录，也欢迎直接将问题回馈给出版社或笔者，除了可以直接回答您的问题外，如果有机会我也可以将其补充在下一版本中，让其他的读者也可以看到。

总结

总体来说，FHS 是 Linux 目录的一个未来方向，也是遵循的标准，虽然 FHS 是一个不错的标准，其目标对所有 Linux 的用户非常好也非常有用，只是各家 Linux 的大厂愿不愿意一起来配合，还有待观察。

根目录的概念非常重要，因为这代表 Linux 和 Windows 的差异（这应该也算是很重要的差别吧☺），在 Linux 下，当还没有 Virtual Machine 的时候，Linux 就已经可以做到一部分切换操作系统的功能，可以用已有的 kernel 来操作其他兼容的操作系统，将会带给用户很多方便性。

另外，也因为根目录的概念，让 Linux 下的目录和分割区的关系变得让普通用户感觉更透明、更有弹性，并且在使用上，除了根目录外，更重要的是根目录下的每一个目录存在的意义，因为都是 Linux 用心分类后的结果。若可以先大概知道每一个目录储存的信息或文件类型，对 Linux 的管理将非常有帮助。我想对一个有经验的 Linux 用户来说，这个比喻应该是很贴切的，这也是整理本书时希望可以帮助读者的地方。

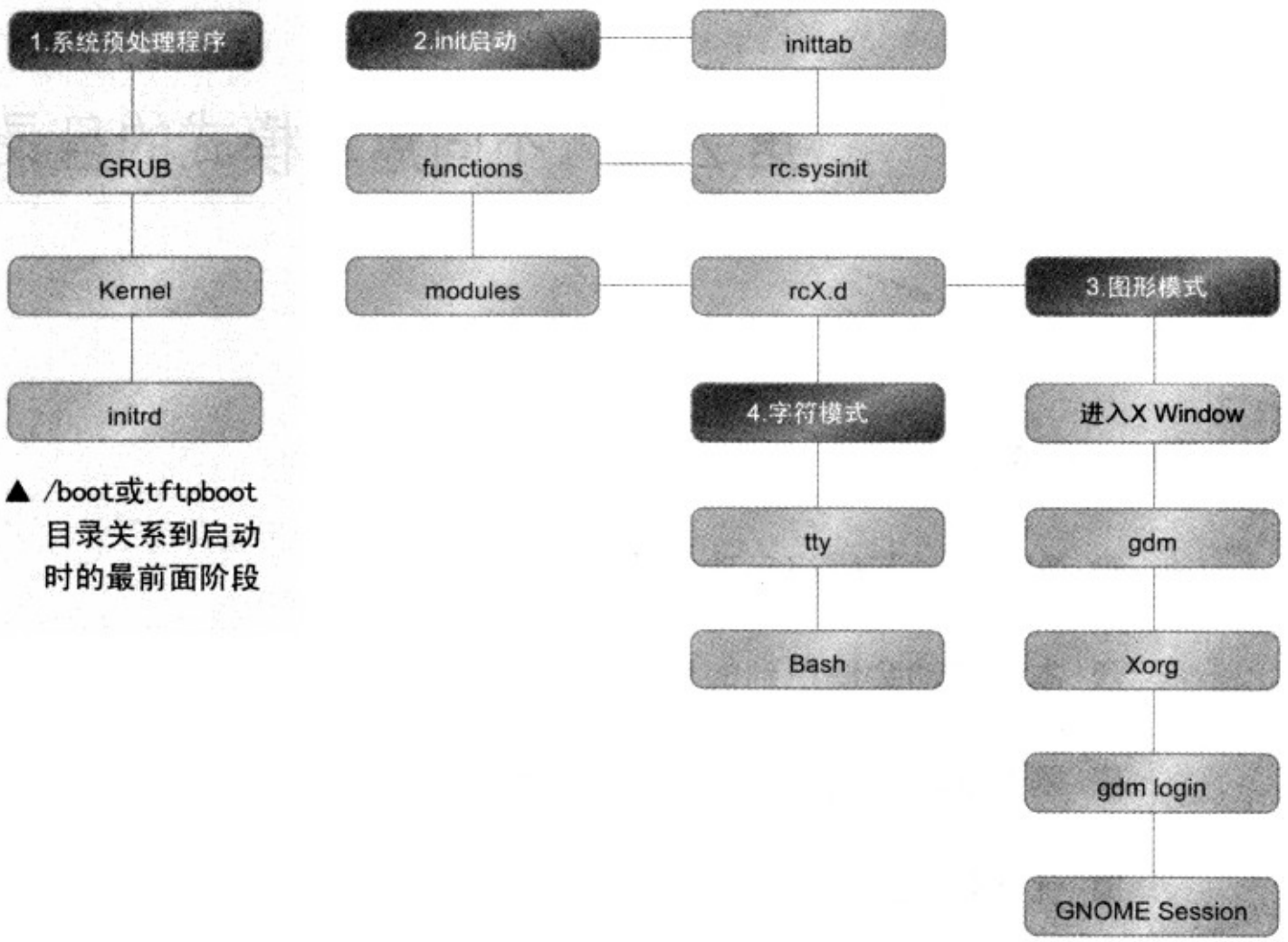
第 2 章　不同启动模式的目录

本章学习重点

- 本地启动与网络启动的差异
- 本地启动会使用到的目录或文件可能会有哪些
- 网络启动会使用到的目录及文件可能会有哪些
- 什么是 System.map 文件及其对 kernel 的意义
- /boot 与/tftp 应用上的差异

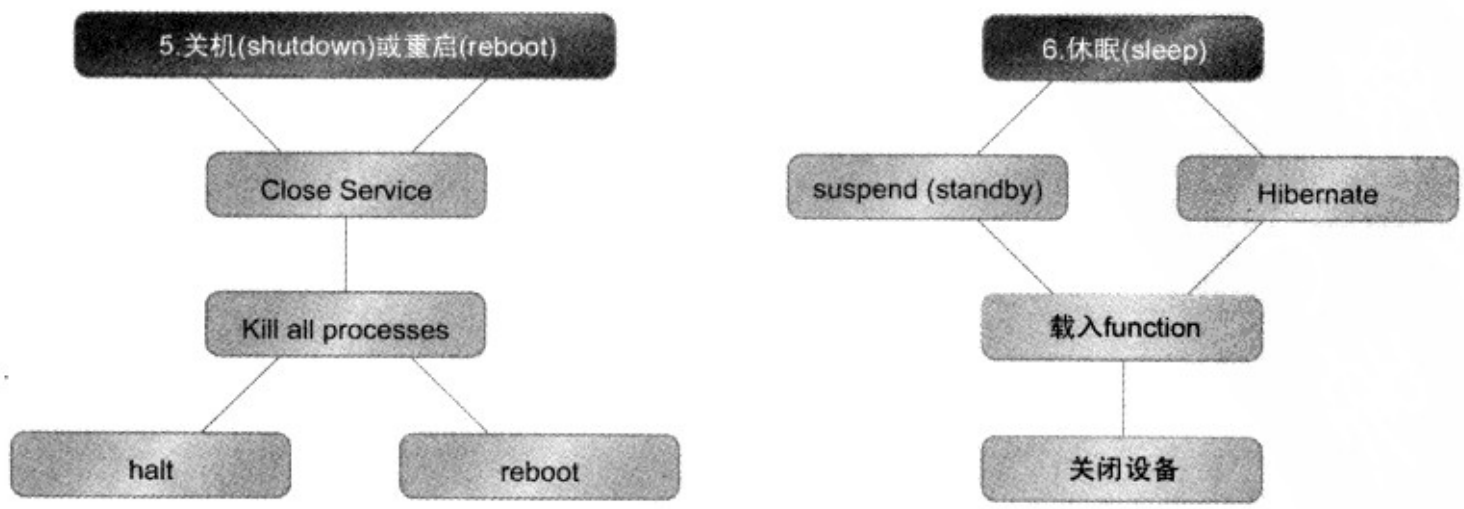
系统流程与章节内容对照图

- 启动流程



▲ /boot或tftpboot 目录关系到启动时的最前面阶段

- 关机流程



各种启动模式在操作系统中都大同小异，不是只有 Linux 如此，所以在此提到的，并不限于 Linux 操作系统；而关于启动方式，在各种实际的操作系统使用上，会因为环境、管理或软件应用而造成差异，像一般单机使用和集群服务器的启动模式就有所区别。一般来说，有两种启动方式：“本地启动”及“网络启动”。

这也是在应用上比较大的不同，本地启动与网络启动的画面也不一样，如图 2-1 与图 2-2 所示，因为本地启动时的启动文件（如 kernel 及 initrd，甚至是系统所使用的系统文件）一定要存在于本地的存储设备（如硬盘）中；而网络启动的启动文件就可以存在远程的某个地点，其优点不只是节省存储空间（启动所需的文件都已放到远程的主机中），也可以因为大量数据是经由内存所执行，可以加速整体的启动速度（当然，如果内存太小，就是自找苦吃了）。

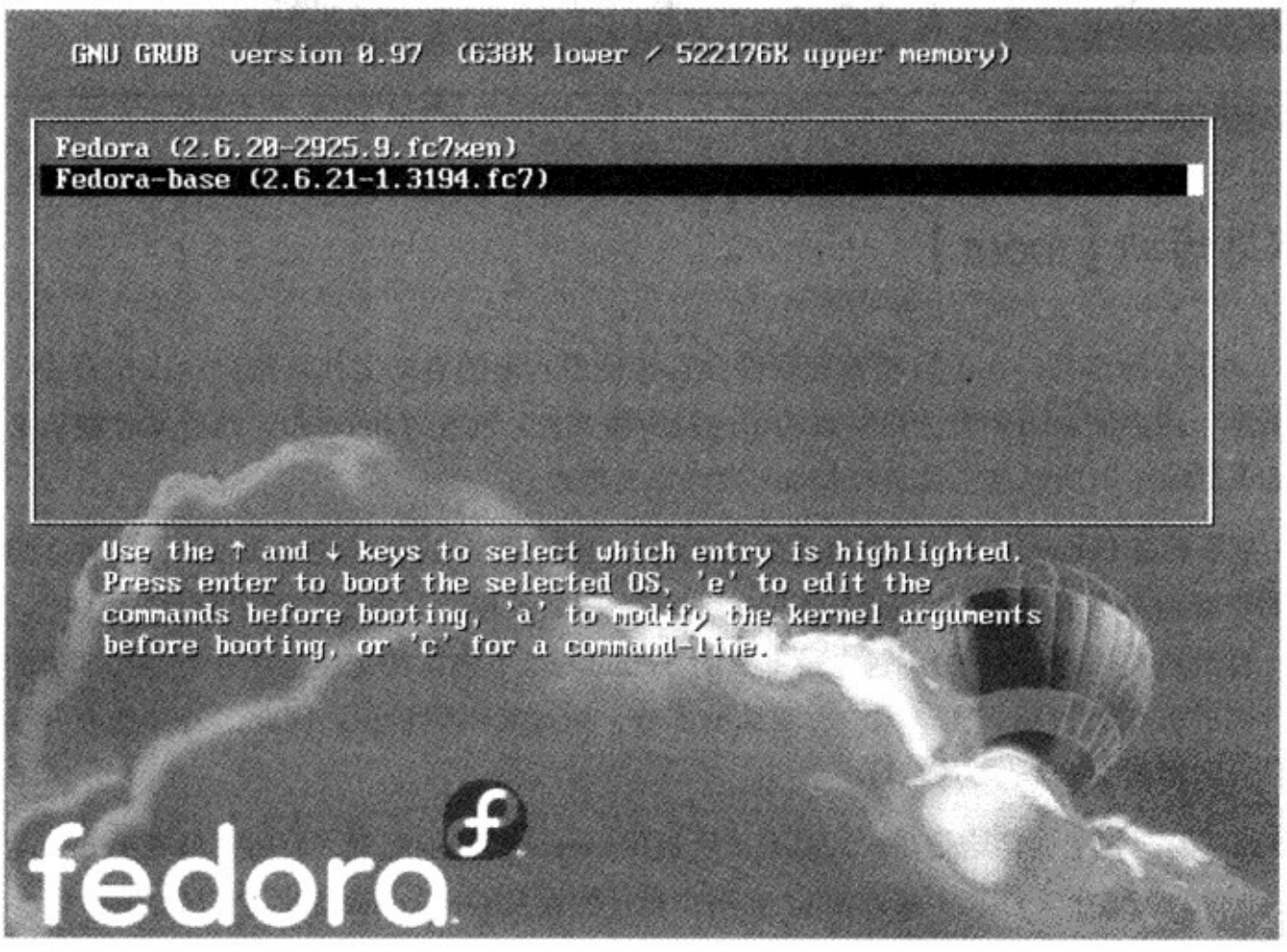


图 2-1：本地启动时的启动画面

因为本地启动与网络启动是由不同的应用程序（本地启动由 GRUB；网络启动由 PXE 及 TFTP）所支持的，所以储存文件的地方就会有差异，为了凸显这一章的特点，本章将依据不同的启动模式，分为两个主要的目录章节来介绍：本地启动【/boot】和远程启动【/tftpboot】。

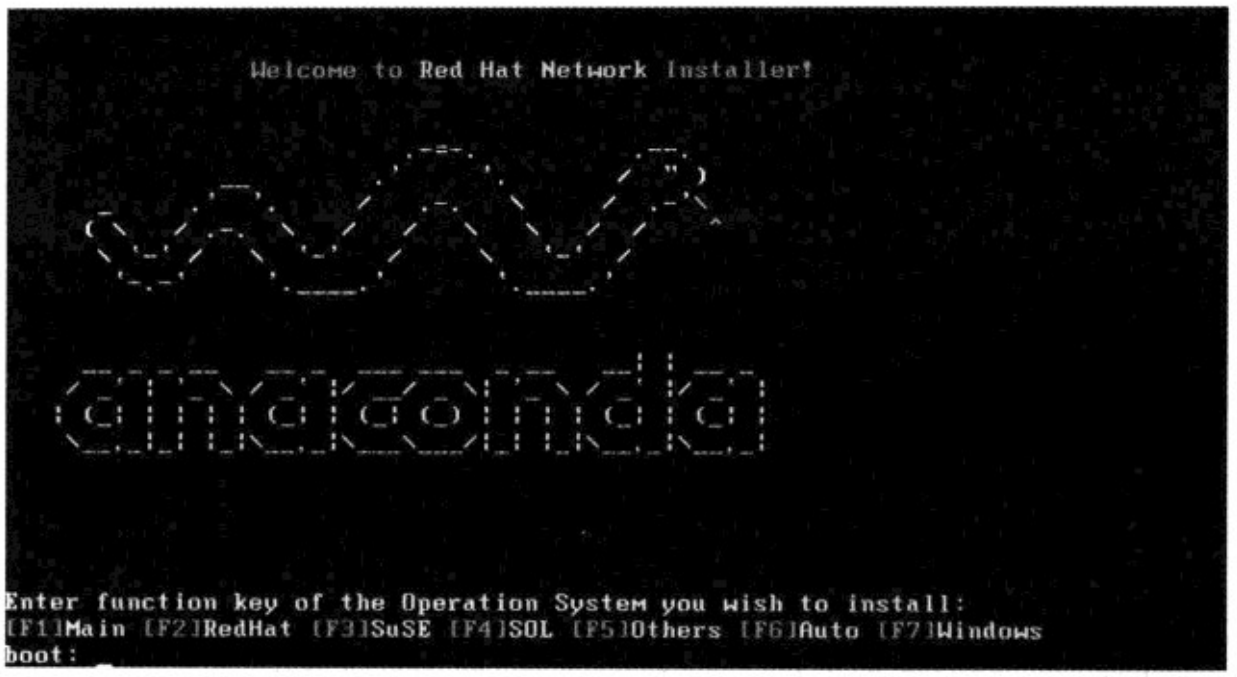


图 2-2：网络启动时的画面

2.1 本地启动【/boot】

本地启动主要的意义，在于启动管理软件会直接使用本地的启动文件直接启动操作系统，所以，在本地一定会储存很多有关启动时的信息及所需文件，而在 Linux 中，这些文件默认会放在【/boot】目录中，所以这个目录对 Linux 来说意义重大。

图 2-3 是笔者在 Fedora 7【/boot】目录中的清单，里面存放各式所需的启动相关的文件（启动时详细的步骤请参考笔者另一本著作《Linux 操作系统之奥秘》，（电子工业出版社，2008），这边以单一目录或文件介绍为主），包含最重要的 kernel，以下将各种文件先行分类（同一类的文件其文件名的开头大致都一样，图中所圈选的部分，怕读者不好辨识，因此，若文件名有重复的部分，只选择其中一个作为标示）。

Ⓐ 系统 kernel 的配置文件，这是在安装时系统所提供 kernel 的功能配置（有时要检查 kernel 功能支持与否，用这个文件其实很方便），在用户自行配置及编译 kernel 后，会以新的取代。

Ⓑ 启动管理程序 GRUB 的目录，里面存放的都是 GRUB 在启动时所需要的画面、配置及各阶段（GRUB 启动时分为 Stage1、Stage1.5 及 Stage2 阶段）的文件，在“第 2.1.1 节：/boot/grub”中有较详细的介绍。

Ⓒ initrd（Initial Ram Disk）文件是系统启动时的模块供应主要来源。

Ⓓ System.map 文件是系统 kernel 中的变量对应表，这个文件在“第 2.1.2 节：System.map 文件”中会有较详细的介绍。

Ⓔ vmlinuz 是在启动过程中最重要的一个文件，因为这一文件就是实际系统所使用的 kernel，若这个文件误删或是名称有误，GRUB 就无法找到 kernel 进行启动（其他的文件都还有启动的机会）。

Ⓕ 这两个文件都是 XEN（支持虚拟操作系统的功能）相关的文件，xen.gz 是当 XEN 在启动时实际会加载的 kernel，而 xen-syms 文件则是在 XEN 问题产生时，可用来 debug 用的对应表（有点像是刚刚所提到的 System.map 文件）。

```
[root@LinuxTree ~]# ls /boot/
config-2.6.20-2925.9.fc7xen  A   System.map-2.6.20-2925.9.fc7xen  D
config-2.6.21-1.3194.fc7         System.map-2.6.21-1.3194.fc7
grub  B                          vmlinuz-2.6.20-2925.9.fc7xen  E
initrd-2.6.20-2925.9.fc7xen.img  vmlinuz-2.6.21-1.3194.fc7
initrd-2.6.21-1.3194.fc7.img     xen.gz-2.6.20-2925.9.fc7  F
lost+found                       xen-syms-2.6.20-2925.9.fc7
[root@LinuxTree ~]#  C
```

图 2-3：/boot 目录中的文件清单

要知道启动时为何会使用到/boot 的目录，首先必须了解在【/boot/grub】中【grub.conf】这个配置文件（每一个操作系统其文件名可能不一样，像在 SuSE 中要找的 menu.lst 这个文件），先看一下在 Fedora 7 中的 grub.conf，以便了解启动时所需要的目录，以及其使用的时间。

图 2-4 是笔者计算机中/boot/grub/grub.conf 文件的内容，这是一个很少见的文件，纯粹是安装时系统自动产生的。

```
[root@localhost ~]# cat /boot/grub/grub.conf
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/VolGroup00/LogVol00
#          initrd /initrd-version.img
#boot=/dev/sda
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz
hiddenmenu
title Fedora (2.6.21-1.3194.fc7)
      A root (hd0,0)
        kernel /vmlinuz-2.6.21-1.3194.fc7 ro root=/dev/VolGroup00/LogVol00
        initrd /initrd-2.6.21-1.3194.fc7.img
[root@localhost ~]#
```

图 2-4：Fedora 中 grub.conf 的范例内容

不过，通过这个文件可以知道为何会使用到/boot 这个目录。在第一个白色方框中，叙述这个配置文件的一部分内容，这一段就是在提醒用户，因为系统中已经存在【/boot】目录，所以启动时所需的 kernel 及 initrd（Initial Ramdisk）文件，其路径应该都要指向/boot 目录中。

也就是说，不论【/boot】是一个根目录下的目录或独立的分区，GRUB 开机管理程序默认都会在启动时，直接指定到/boot 的路径下，所以，在图中标示的【root (hd0,0)】字符串，其实代表/boot 的路径，至于接下来的【kernel /】及【initrd /】，并不是指到一般的【/】根目录，而是要接着【root (hd0,0)】的/boot 目录。

因此【kernel /】及【initrd /】的意思其实是【kernel /boot/】及【initrd/boot/】，所以 GRUB 在找 kernel 及 initrd 文件时，是在/boot 中去寻找，而不是在根目录下寻找。

这样说或许还不是很清楚，简单地说，根目录和/boot 启动用的目录是完全没关系的，例如，如果有 5 个操作系统，分别放在 5 个分区中，这 5 个操作系统的启动用 kernel 及 initrd 文件，可以全部放在第 6 个分区。

如图 2-5 中的 A、B、C 三部分（关于 B、C 两部分的启动配置信息，在本章“第 2.1.1 节：/boot/grub”中会有更详细的介绍）。

Ⓐ 笔者将启动用的 kernel 及 initrd 两个文件，放到操作系统外的另一个【/dev/sdb1】分区中。

Ⓑ 再将【grub.conf】中的第一段 root 记录由原本的【(hd0,0)】改为【hd(1,0)】，代表从第一块硬盘变为第二块硬盘。

Ⓒ 系统到底在哪里？其实系统的目录是在 kernel 的启动参数中，所以和前面所看到启动时所用的【/boot】是完全没有关系的，只是大部分的系统因为【/boot】目录都没有另行设计过，都会将 B 的 root 与 C 的 root 误以为是同一件事，其实不然。

这样在启动时，GRUB 启动管理程序一开始就会读取原本【/dev/sda】中的/boot/grub 目录来启动画面，因为这部分是直接记录在引导扇区中的。等真正要通过启动文件启动时，GRUB 就会按照配置文件，将 kernel 及 initrd 文件从【/dev/sdb】分区中加载，而不是从原本的【/dev/sda】中读取。

```
[root@localhost ~]# cat /proc/partitions
major minor  #blocks  name

   8     0   10485760 sda
   8     1     104391 sda1
   8     2   10377990 sda2
   8    16     524288 sdb      A
   8    17     522081 sdb1
 253     0    9273344 dm-0
 253     1    1048576 dm-1
[root@localhost ~]# tail -4 /boot/grub/grub.conf
title Fedora (2.6.21-1.3194.fc7)
        root (hd1,0)  B
        kernel /vmlinuz-2.6.21-1.3194.fc7 ro root=/dev/VolGroup00/LogVol00  C
        initrd /initrd-2.6.21-1.3194.fc7.img
[root@localhost ~]#
```

图 2-5：将启动用的目录改到别的分区中

假设有 5 个操作系统，可以将 5 个操作系统所须使用的 kernel 及 initrd 文件，全部放到某一

个特定的分区中，这样在多重操作系统的管理上，会比较方便，也比较好做启动备份。

2.1.1 /boot/grub

这个目录和“系统”启动其实没有太大关系，从启动顺序来说，反而是在系统启动之前，因为这里所存放的是启动管理程序所需的文件及程序，如一般常见的【grub.conf】配置文件就是在这个目录中。

所以，在整个启动过程中，可以看到和/boot/grub 相关的部分，大概就是像图 2-6 中启动管理软件所呈现的画面及其中的配置。

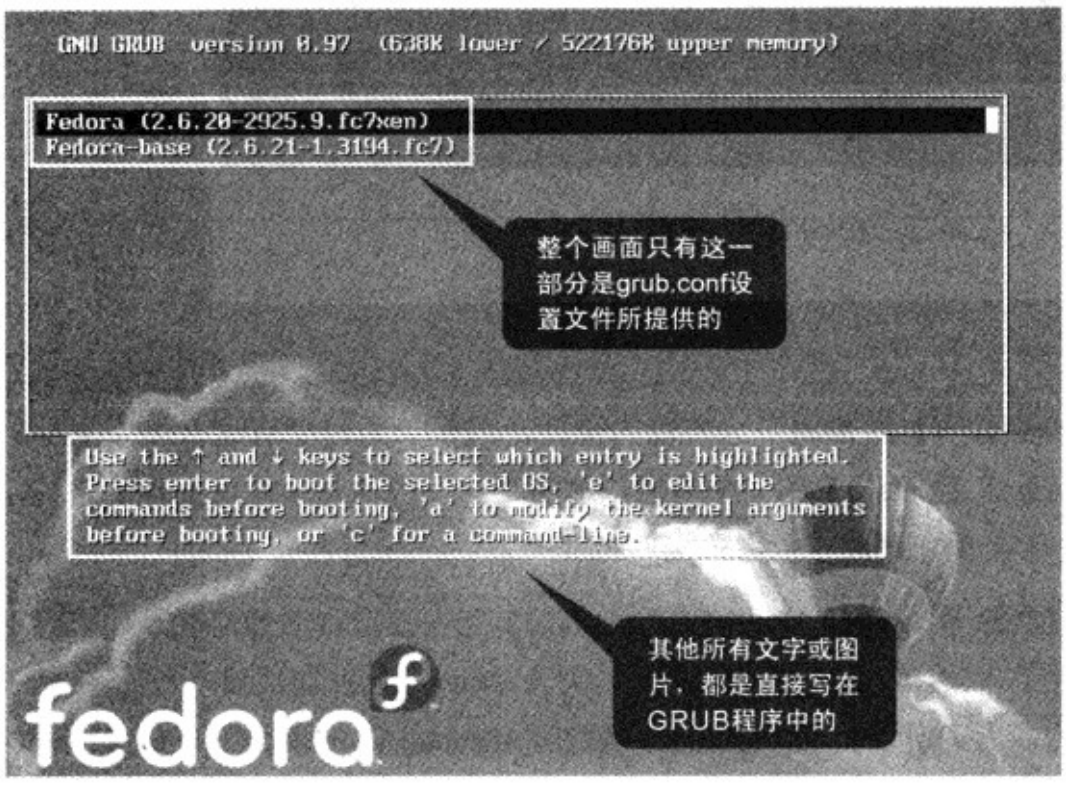


图 2-6：GRUB 启动画面

由此可知，在【/boot/grub】中的文件和启动时的画面（呈现效果或功能）是紧密结合在一起的，因此，每一个文件都有其特殊的功能。这里将所有文件（如图 2-7 所示）依不同属性整理为以下几类。

```
[root@LinuxTree grub]# ls
device.map      grub.conf          minix_stage1_5     stage2
e2fs_stage1_5   iso9660_stage1_5   reiserfs_stage1_5  ufs2_stage1_5
fat_stage1_5    jfs_stage1_5       splash.xpm.gz      vstafs_stage1_5
ffs_stage1_5    menu.lst           stage1             xfs_stage1_5
[root@LinuxTree grub]#
```

图 2-7：/boot/grub 目录的文件清单

◆ Stage1 阶段

也就是 stage1 文件，名称刚好取得一模一样，读者可以想成是硬盘引导扇区（MBR）的备份文件，但是这个备份文件是在安装前所做的，所以中间的分区部分（其中的 64bytes）是有问题的，如果要备份，最好还是将 stage1 以正确的 MBR 重新备份一次。

基本上，stage1 只负责做引导的动作，不会有太多的功能，毕竟它也只有 512bytes 的大小（真正的 stage1 其实只能算 446bytes），如果用户在启动时可以看到 GRUB Shell，那就是 stage1 的真正画面（如图 2-8 所示，不过通常会看到 GRUB Shell 并不是一个好现象）。

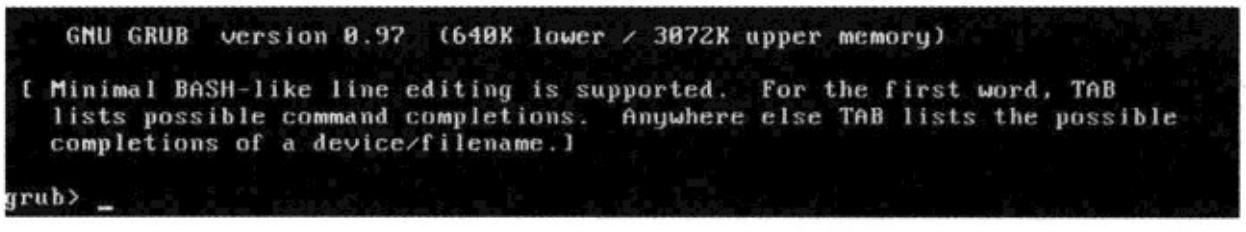


图 2-8：GRUB Shell 的画面

◆ Stage1.5 阶段

所有文件名是【XXX_stage1_5】的都属于这阶段的文件，但至于为何会有这么多个文件？是因为其作用就像是连接 stage1 到 stage2 的一个通道，里面唯一存放的是该系统文件的格式，所以只要是被支持的格式，就会预先存放一个格式文件在其中。

这阶段的过程可以让 GRUB 在 stage1 启动完成后，stage2 能在被搬移后的情况下，就算不在原本的目录或文件系统中，依然可以被安全地找到。因为 stage1.5 被加载时，就已经赋予 GRUB 读取文件系统目录的能力，所以自然可以在一开始找不到 stage2 的情况下，从文件系统目录中，找出 stage2 的所在位置。

不过，通常 stage1.5 阶段的文件不会放在目录中，因为当 stage1 还没加载 stage1.5 时，原则上是不能识别 ext2 的，当然也无法找到 stage1.5 这个文件，所以，其实 stage1.5 是存在硬盘最前面的 32KB 的区段中的（但是要跳过 MBR），当 stage1 调用 stage1.5 时，就直接去该区域将 stage1.5 找出来使用。

◆ Stage2 阶段

文件名也是叫 stage2，该文件是 GRUB 的核心程序，能让用户以菜单方式将操作系统加载、新增参数、修改选项，这些全都是 stage2 的功用。对 GRUB 来说，stage2 除了不能自己启动外，剩下的事情全都由 stage2 完成。像是用户在启动时所看到的 GRUB 启动倒数画面，或是紧接着的启动菜单画面，就都是由 stage2 所提供的。

Stage2 文件主要提供的功能如下。

- 提供菜单。
- 读取配置文件。
- 连接下一个 boot sector。

◆ 配置文件

device.map、menu.lst 及 grub.conf 都属于这类文件。

- device.map 是直接侦测目前的硬件来假设 BIOS 所记录的实体磁盘有哪些，默认值是安装系统时就记录好的。
- menu.lst 在此只是一个链接文件，链接到 grub.conf 文件，但在 SuSE 中却刚好相反，会以 menu.lst 为主要文件。
- grub.conf 就是 GRUB 启动程序的主要配置文件，如果此文件丢失，会让 GRUB 在启动时失去默认的选项，但要强调的是，此时还可以临时（启动时）通过手动配置方式启动（只要其他文件还在）。
- splash.xpm.gz 文件就是启动时的背景图片。

2.1.2 System.map 文件

另外在/boot 目录中，较少听到但值得一提的是 System.map 这个文件（如图 2-9 所示），看文件名就知道它是一个像地图（map）一般的文件，不过，正确的说法应该是索引文件，因为它存在的目的是让外部软件可以知道 kernel 文件内部实际分配的位置。

```
[root@LinuxTree proc]# ls /boot/System.map-2.6.2*
/boot/System.map-2.6.20-2925.9.fc7xen  /boot/System.map-2.6.21-1.3194.fc7
[root@LinuxTree proc]# _
```

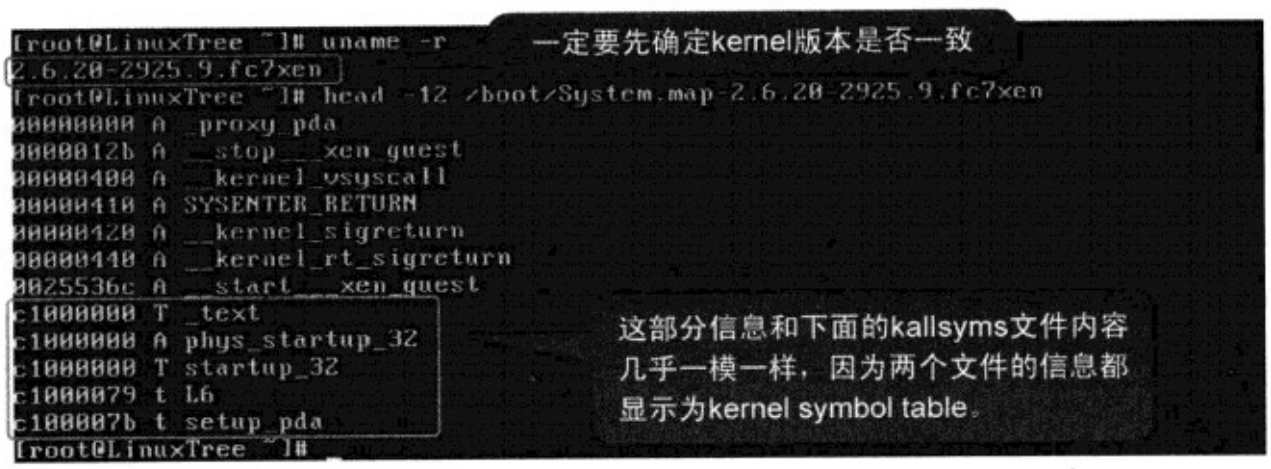
图 2-9：/boot 目录中的 system.map 文件

最常会遇到的是更改 System.map 文件，当用户在重新编译 kernel 时，因为新的 kernel 和原本的 kernel 一定会有所差异，因此，内部的分配也不一样，须重新制作出一个新的 System.map 文件以符合要求。

而这文件的内容是在索引什么东西？答案就是 Symbols。什么是 Symbol？其实就是 kernel 中的变量（Variable Name）或函数名称（Function Name），这样可以方便程序员在写程序时可以直接参照这一份 Symbol 的索引文件，找到所需要的 kernel 信息，这一份 Symbol 的索引文件又

称为 kernel symbol table。

这样说好像很抽象，看实际的文件会比较有感觉。目前较常使用到的 kernelsymbol table 文件有两个（实际上会用到的不止这些，在此只是举例），即/proc/kallsyms 和这个章节提到的 system.map 文件。System.map 文件较单纯，是在用户一开始编译就产生的固定文件，不会因为任何原因更改，除非被换掉。而/proc/kallsyms 是一个在启动时由 Linux kernel 实时产生的文件，当系统有任何变更时，它就会马上做出修正。两者的差异点如图 2-10、图 2-11 所示。




```
[root@LinuxTree ~]# uname -r
2.6.20-2925.9.fc7xen
[root@LinuxTree ~]# head -12 /boot/System.map-2.6.20-2925.9.fc7xen
00000000 A __proxy_pda
0000012b A __stop___xen_guest
00000400 A __kernel_vsyscall
00000410 A SYSENTER_RETURN
00000420 A __kernel_sigreturn
00000440 A __kernel_rt_sigreturn
0025536c A __start___xen_guest
c1000000 T _text
c1000000 A phys_startup_32
c1000000 T startup_32
c1000079 t L6
c100007b t setup_pda
[root@LinuxTree ~]#
```

图 2-10：System.map 文件的前 10 笔记录

```
[root@LinuxTree ~]# uname -r
2.6.20-2925.9.fc7xen
[root@LinuxTree ~]# head -4 /proc/kallsyms
c1000000 T _text
c1000000 T startup_32
c1000079 t L6
c100007b t setup_pda
[root@LinuxTree ~]#
```

图 2-11：Kallsyms 文件的前 4 笔记录

从图 2-10、图 2-11 可以看出，两个文件的前面部分（其实到后面也是如此）是几乎完全一致的，因为两份数据其实都是为了将目前所使用的 linux kernel 中的 symbol 信息呈现给用户，但为何在 System.map 的前面一小部分会有些许的不同？笔者并没有找到相关的资料，**但斗胆猜测，请注意，只是笔者的猜测唷！**应该是因为 System.map 中的信息是 kernel 的完整数据，也就是说，从一开始载入 kernel 就已经是照表运行；但/proc/kallsyms 则不是，它是动态产生，换句话说，是 kernel 进行到一半才开始制作出来的。所以这小部分的差异，笔者认为可能是在 kernel 加载一直到产生出/proc/kallsyms 这段时间所进行的部分，因此，无法写入/proc/kallsyms 中。

但是否因为如此，就表示 System.map 的内容比/proc/kallsyms 来得多，如果这样以为就大错特错了，应该说，/proc/kallsyms 的信息比 System.map 多太多了，我们看一下这两份文件的尾部就可以知道差异在哪（如图 2-12、图 2-13 所示）。

在固定文件的 System.map 中，pg0 是最后一条 Symbol 数据，Kernel 中所记录的位置为 c15e5000；对应到动态产生的/proc/kallsyms 文件中的 pg0，同样指到 kernel 的 c15e5000 地址。但注意到/proc/kallsyms 接下来的内容是原本 System.map 中所没有的，这些是哪里来的呢？这些都是系统中所使用到的 modules，像网卡、USB、EXT3 文件系统等的 modules，这是不是对一般用户来说更有意义？因为可以通过这文件，直接知道目前所有 kernel 或 module 使用 Symbol 的状况（当然这对笔者影响极小，因为笔者不在写程序 ☺）。

```
[root@LinuxTree ~]# tail -5 /boot/System.map-2.6.20-2925.9.fc7xen
c15e484c b netlabel_unlabel_acceptflg
c15e4850 b __key.11080
c15e4858 B __bss_stop
c15e4858 B _end
c15e5000 B pg0
[root@LinuxTree ~]#
```

图 2-12：System.map 的后 5 笔记录

```
c15e484c b netlabel_unlabel_acceptflg
c15e4850 b __key.11080
c15e4858 B __bss_stop
c15e4858 B _end
c15e5000 B pg0
debfb00a t tx_add_credit        [netbk]
debfb052 t make_rx_response     [netbk]
debfb117 t make_tx_response     [netbk]
```

图 2-13：/proc/kallsyms 文件中的一部分记录

笔者要特别强调，因为这是动态的信息，当用户新增或删除一个 module，都会自动做实时的修正（/proc 下的都是这一类型的文件），像载入 USB module 时所有使用到的 Symbol 都可以实时看到（如图 2-14 所示）。

```
[root@LinuxTree ~]# grep usb_storage /proc/kallsyms
[root@LinuxTree ~]# modprobe usb_storage
[root@LinuxTree ~]# grep usb_storage /proc/kallsyms | head -5
dec25000 t host_info    [usb_storage]
dec2500a t store_max_sectors    [usb_storage]
dec25063 t show_max_sectors     [usb_storage]
dec2508b t proc_info    [usb_storage]
dec2537f t slave_alloc  [usb_storage]
[root@LinuxTree ~]#
```

图 2-14：USB module 对/proc/kallsyms 的影响

2.1.3 kernel 及 initrd

kernel 就是我们刚刚所提到的“vmlinuz”文件，只是文件名称不是直接取名为 kernel 罢了。一般系统在启动时，kernel 及 initrd 两种文件都是要同时存在的（如图 2-15 所示，启动菜单中大部分都是 kenel 与 initrd 两个选项同时存在的），因为这两者是相辅相成的文件，缺一不

可，这也是为何笔者会将 kernel 及 initrd 放在同一章节中做介绍的原因，这样可以让读者直接将两者做对照。当然，不同的系统应用模式或许可以省略掉 initrd，但 kernel 就万万不可，因为 kernel 是一个系统（不论系统的大小）中最基本的一个元素，缺了 kernel，就无系统可言。

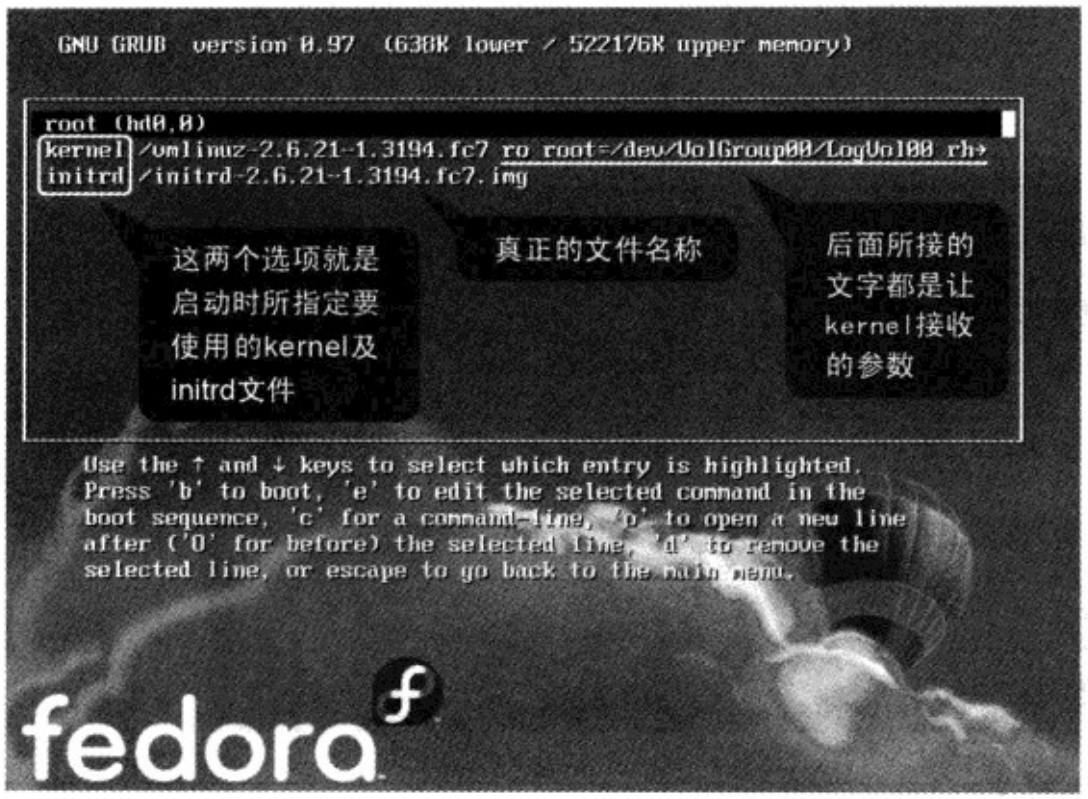


图 2-15：GRUB 启动菜单中某操作系统的细部选项

◆ kernel

kernel 是操作系统的核心程序，也是操作系统最重要的一支程序，所有的软、硬件都通过 kernel 做交互的操作，因此，若这段程序写得不好，将会造成操作系统易宕机的状况，甚至找不到设备或适配器都有可能。

用户和 kernel 之间，最常会遇到的一种状况，就是当计算机使用一段时间（可能是 2~3 年内），就会发现许多硬件或软件功能都已经不再被既有的操作系统所支持，像 USB 2.0 刚出现或是须要使用新的文件系统（kernel 中有关 USB 2.0 的配置如图 2-16 所示），都要通过 kernel 的更新做配合，因为软、硬件技术都已经全面更新。

【config-kernel 版本】文件的意义，就在于记录 kernel 所支持的项目有哪些，以及支持的方式（像直接支持或是模块化），当然每一版 kernel 的 config 文件都会不太一样。

举例来说，近几年的 Intel 南桥芯片组依序为 ICH5、ICH6、ICH7、ICH8、ICH9，一直到现在的 ICH10，如果用户在 ICH10 刚推出就安装 Fedora 7，启动时一定会有许多兼容性的问题发生，因为芯片组有可能部分功能不被 Fedora 7 支持所致。

```
[root@LinuxTree boot]# grep -i ehci config-2.6.21-1.3194.fc7
CONFIG_USB_ARCH_HAS_EHCI=y
CONFIG_USB_EHCI_HCD=m
CONFIG_USB_EHCI_SPLIT_ISO=y
CONFIG_USB_EHCI_ROOT_HUB_TT=y
CONFIG_USB_EHCI_TT_NEWSCHED=y
# CONFIG_USB_EHCI_BIG_ENDIAN_MMIO is not set
[root@LinuxTree boot]# grep -i ntfs config-2.6.21-1.3194.fc7
# CONFIG_NTFS_FS is not set
[root@LinuxTree boot]#
```

图 2-16：在/boot/config-XXX 文件中有关 USB 2.0 与 NTFS 的配置

解决的方法，并不是一定得苦等到更新版本的出现，而是可以在既有的 Fedora 7 直接从网络上下载较新的 kernel 版本，再将旧的 kernel 取代掉即可，只要新的 kernel 支持该芯片组，便可以继续使用原来的 Linux，完全不需重新安装或增删其他任何资料。

◆ initrd

initrd 的全名是 initial ram disk，顾名思义就是启动系统所需加载的虚拟磁盘。要了解 kernel 与 initrd 之关系，才会真正知道 initrd 存在的意义及它执行时对系统所带来的影响，这是非常重要的。

以图 2-17 为例，kernel 、initrd 与系统硬盘中所存有的模块（System Module）有各自所负责的领域，这三者依启动时的顺序被依序加载，因为启动过程需要很多硬件的协助，所以在顺序上是不能混乱的，在图 2-17 中为了让读者比较好分辨出各硬件与这三者的关系，所以用不同

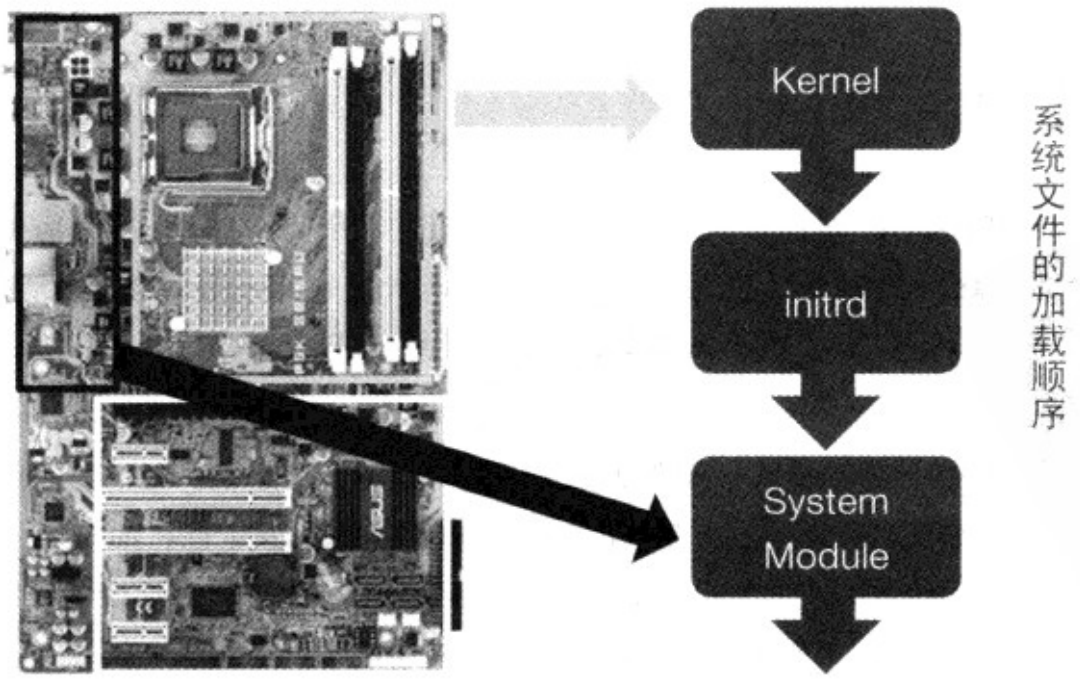


图 2-17：kernel、initrd 与模块间的关系

的颜色深浅来标明（大致的范围，仅供参考使用），同颜色箭头所指向的系统文件负责该区域，这三个系统文件依启动顺序分别为：kernel、initrd 及 System Module，以下分别介绍其负责的部分有哪些。

- 灰色框：kernel 所负责的主要是北桥、南桥、CPU 及内存，从硬件角度来看，若 kernel 有问题，当然是整个系统都会出错，因为会影响到整部主机最重要的硬件核心部分。
- 白色框：initrd 大部分所支持的都是一些关键（会影响启动）的外部硬件，如 SATA、SCSI、USB 等硬件外设，可视情况调整支持的多少。
- 黑色框：系统中所存在的模块，是与支持和启动无很大关系的硬件，大部分是一些应用方面的硬件设备，如声卡、网卡、显示卡等，没有这些硬件的支持还是可以正常启动进入 Linux，只是会有功能上的缺陷，所以才会以模块文件方式存在于文件系统中。

initrd 的目的就是在 kernel 加载系统认识 CPU、内存等信息之后，像接力赛般地让系统进一步知道**“还有哪些硬件是启动所必须使用到的”**，若和启动无关的，就可以丢给接下来用户自行配置的系统模块。

但什么时候需要在 initrd 文件中放入额外的模块（或拿掉模块），答案是当主机多了一个在启动操作系统之前就要先找到的设备。

如果该硬设备可以等进入操作系统再使用，就可以直接以 modprobe 的方式载入模块使用，而无须去修改 initrd 文件，因为这样只会增加启动时所要加载的文件数目。但要注意的是，**在 initrd 文件中所放入的模块，必须是与操作系统同一版本 kernel 所编译的模块，这样才可使用。**

2.2 远程启动【/tftpboot】

这个目录必须要安装 tftp-server 组件才会产生，不然是没有的，因为并非基本操作所使用。一般网管人员经常会遇到这一个目录，因为在大部分的 switch、router 等设备需要更新其 firmware（固件）时，都会支持以 TFTP 的方式做更新，而不是用 FTP 或 Web，因为其协议较简易，也无需安全性的考虑。

TFTP 是一种相当简单的文件传输模式（如图 2-18 所示），和 FTP 比起来，最精简的部分就是不需要账号、密码即可传输文件。基本上，TFTP 的目的就不是为了安全性，而是为了方便性。像图 2-18 就是 TFTP 实际传输的一个例子，用户要从【192.4.1.161】主机上通过 TFTP 下载一个【file1】文件，只须执行一道简单的指令，无需任何验证动作，就可以完成下载。

在 TFTP Server 的部分，也只须将【file1】文件先行放到【/tftpboot】目录中即可，当用户要从 Server 下载该文件时，TFTP 会自动以此目录为主目录，直接传送给远程的用户。

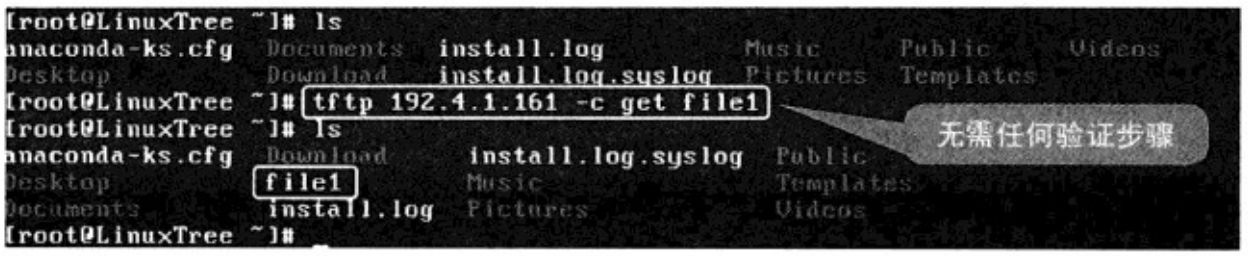


图 2-18：TFTP 的作业方式范例

更新 firmware 是一种 TFTP 用法，另一种也经常看到的（最近越来越多）就是远程启动。正确地说，是通过 TFTP 的方式将 kernel 加载，经由网卡加载到本地内存中执行，也可以说是无盘启动（Diskless Boot）模式，较常看到的应用为 clustering 技术或是 PXE（Preboot eXcution Environment）安装模式。

以 PXE 启动方式（如图 2-19 所示）来介绍【/tftpboot】目录应该会比较清楚，简单地说，PXE 就是 TFTP 加上 DHCP 的一种应用方式（如图 2-19 所示），但当然不是真的这么简单，还要 BIOS 及网卡的固件双方都支持 PXE 的功能，不过以现在的软硬件，应该都已经具备这些功能了。

在图 2-19 的 6 个步骤中（这里的步骤只列到可以启动，未含完整安装步骤），在第 3 步骤，用户会向 TFTP Server 请求一个简单的“启动菜单”，供用户选择以哪一个操作系统启动。而第 4 步骤就是第一次和【/tftpboot】目录有直接关系的地方，因为在这一步骤中，TFTP Server 就会将【/tftpboot】目录中存有“启动菜单”的文件直接传送给远程的用户，让用户有可启动的菜单画面。

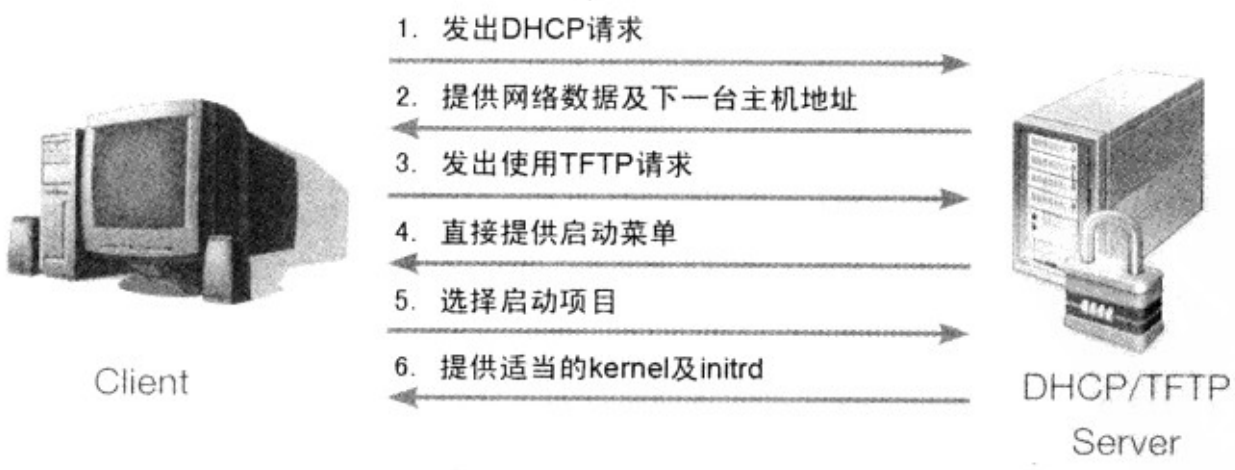


图 2-19：PXE 启动的概念图

对这种方式感兴趣的读者，可以参考 syslinux 的网站：

http://www.syslinux.com

它提供现成的 PXE 所需组件（一般是可用现成的组件做到此功能）及安装方式，只要配合 DHCP 的使用，就可以做到 PXE 安装，本书就不再赘述服务的安装方式。如有进行无盘启动应用的需求，则必须进一步学习 embeddedlinux 的技术，因为无盘的目的就是要让系统在不使用存储器的状态下进入系统，所以，启动所用的系统文件（如 kernel 与 initrd）都不会太大，大都是以 embedded Linux 为主。

总结

不论是/boot 或/tftboot 目录，其存在的目的都是为了系统启动，换言之，如果某一个操作系统不需要自行启动（可以通过别的系统启动），这两个目录存在的意义就不大（单指本地而言），因为本身已经不需要启动文件，唯一有可能会被系统使用到的文件，就是 System.map。

但不管如何，启动是一个系统最基本的功能，/boot 目录包含最基本的启动管理系统所需要的所有文件，也储存着最重要的 kernel 及 initrd 文件；而远程启动所要用到的/tftpboot 目录，同样是为了放置让远程启动所使用的启动系统文件，只是提供的方式和/boot 将有所差异，但因为通过网络增加了许多的方便性，在未来的 Linux 世界中，这一类的应用一定会越来越广泛。

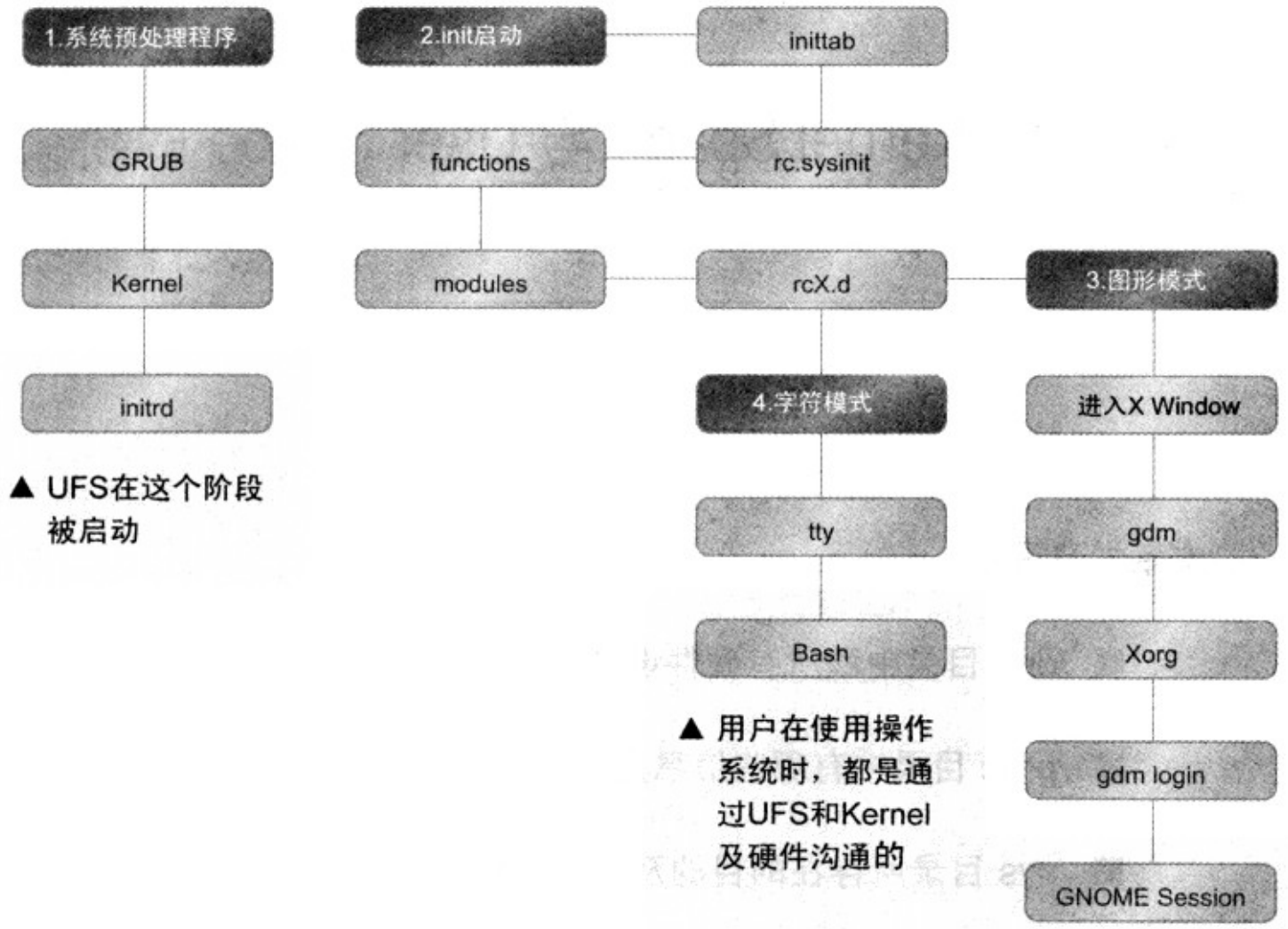
第 3 章　Kernel Space 与 User Space 的桥梁
——虚拟文件系统

本章学习重点

- **/dev 目录中和所有硬件设备的关联**
- **/proc 目录中有哪些信息是可以参考而平时没注意到的**
- **/sys 目录所存在的目的及其方便性**
- **/proc 与/sys 之间的差异**
- **思考是否可以利用 kernel space 的相关目录，带来使用习惯的改进**
- **什么是虚拟目录**

系统流程与章节内容对照图

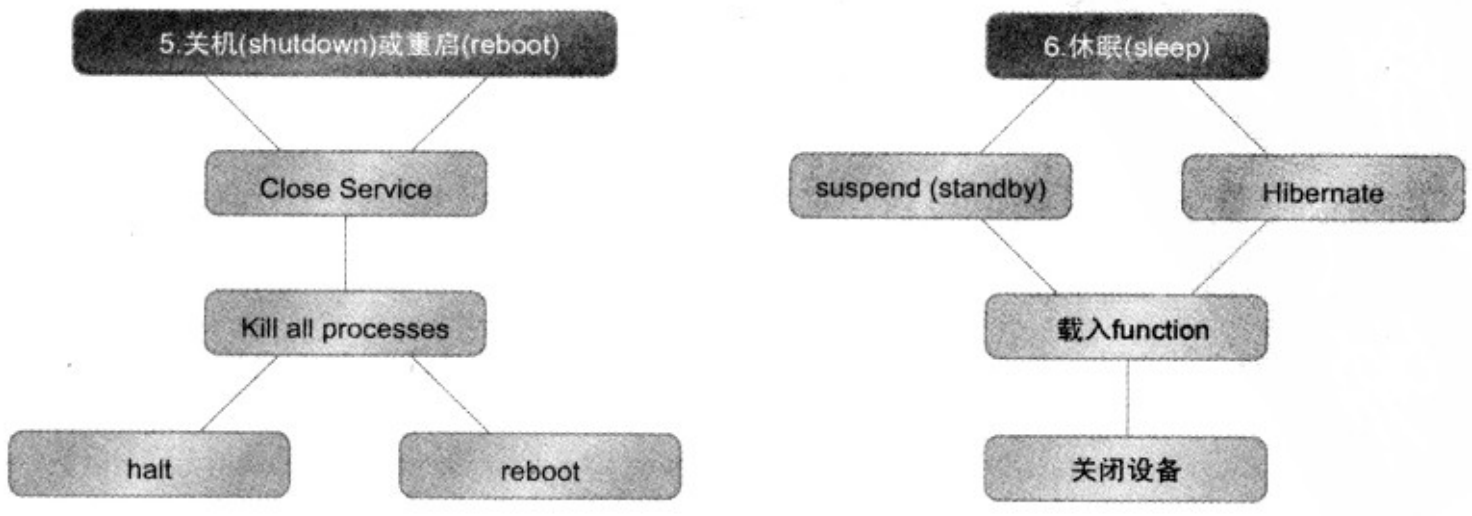
- 启动流程



▲ UFS在这个阶段被启动

▲ 用户在使用操作系统时，都是通过UFS和Kernel及硬件沟通的

- 关机流程



什么是虚拟文件系统？在了解之前，必须要先知道什么是 Kernel Space 与 User Space。**Kernel Space 与 User Space 的差别，在于内存使用上安全机制的差异**，弄清楚这一点，在虚拟文件系统的概念上就可以有比较清晰的理解。

kernel 执行时会占据一段系统的内存空间（如图 3-1 所示），这一段空间便是 Kernel Space，所有用户是无法和 kernel space 直接交互的。被保护是因为操作系统最主要的核心就是在这一个内存区域中持续运行，并提供其运行中的服务与功能，也因为是被保护在内存之中的，其目的除了将系统与用户隔离开来，速度也是很主要的原因，因为这样可以避免在相对“慢速”的硬盘上执行。

```
[root@LinuxTree ~]# head -10 /proc/iomem
00000000-0009f7ff : System RAM
  00000000-00000000 : Crash kernel
0009f800-0009ffff : reserved
000a0000-000bffff : Video RAM area
000c0000-000c7fff : Video ROM
000c8000-000c8fff : Adapter ROM
000f0000-000fffff : System ROM
00100000-1feeffff : System RAM
  00400000-00604a49 : Kernel code
  00604a4a-00715cb3 : Kernel data
[root@LinuxTree ~]#
```

图 3-1：在/proc/iomem 文件中所记录的 kernel 地址

而 User Space 则是强调，用户执行任何的程序所占用的内存部分，对这些 User Space 的程序而言，无法直接使用 Kernel Space 中的资源，必须要经过一些系统所提供的【system calls】才可使用 Kernel Space 的对象。

这也是操作系统中 Ring 的主要概念（如图 3-2 所示），整个操作系统分为 Ring 0、Ring 1、Ring 2 及 Ring 3，Ring 0 是 kernel 所在的地方（也就是 Kernel Space），也是唯一可以和硬件直接交互的系统组件。所有 Ring 0 以外的软件，若须使用到硬件时，都必须通过 Ring 0 的执行。

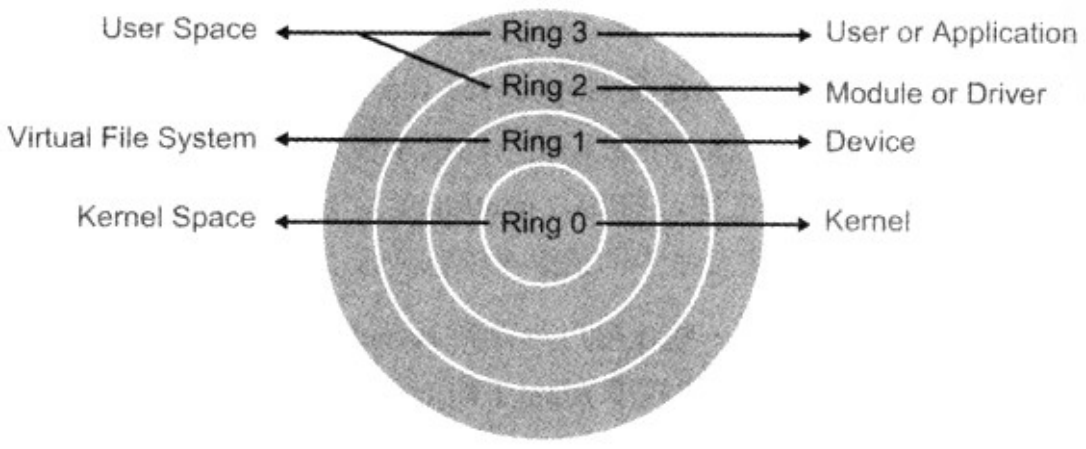


图 3-2：Ring 的概念

而 Ring 1 和 Ring 2 则是系统模块的部分，对用户来说，就是经常看到的一些模块或是本章所要介绍的【/dev】、【/proc】和【/sys】等“虚拟文件系统”（Virtual File System, VFS）。所有用户正常使用的应用程序（也就是 User Space），则是被限制在 Ring 3，其目的就是让应用程序与 kernel 区分开，不然，任何用户都可以使用到该应用程序，等于将 kernel 让所有人直接使用，非常容易产生问题，以这种隔离的方式，让发生错误时的影响降低。因此，当应用程序需要任何的硬件设备时，要先通过虚拟文件系统或系统模块，向 kernel 请求支持。

用户如果在正常的使用环境下，想得知一些 Kernel Space 的信息，像 kernel 的变量、设备的模块信息、硬件的底层信息等，必须通过中间的虚拟文件系统——User Space 与 Kernel Space 的信息才可相互交互。

虚拟文件系统与一般的文件系统差别最大的地方在于“虚拟”二字（好像有点废话），因为只要是虚拟文件系统，就没有一定要在哪一个目录下查看的限制，并且目录下的任何文件都不会占用硬盘的空间，因为虚拟文件系统只是一个抽象的对应方式，并没有任何实体的文件存在硬盘中。

所以，用户千万不要认为一些常见的目录，像【/proc】、【/sys】、【/tmp】等一定都会在 Linux 操作系统下看到，有些默认系统就不一定会直接替用户预先挂载这类目录，因此，当用户要查看这些目录时，就必须自己去找！

在 Linux 的使用上，虚拟文件系统一直是被关注的部分，因为这些都是比较偏 Linux 底层的信息，只要可以掌握这些信息，就比较能够掌握 Linux（当然有心的话，直接掌握 Linux kernel 更好）。而在所有的 Linux 目录中，有很多就是因为 Linux 底层的运行所产生出来的目录及文件，因为原本 Kernel Space 是一般用户无法碰触的地带，如果这些虚拟文件系统下的目录或文件运用得好，对 Linux 的管理、应用或操作，都会带来很大的便利性。

3.1　设备文件目录【/dev】

/dev 目录是由 Linux 的 devfs 所建立出来的，devfs 全名为 device file system，其主要目的在于用来存放所有系统中 device（设备）的相关信息，不论是使用的或未使用的设备，只要有可能使用到，就会在/dev 中建立一个相对应的设备文件，如/dev/input/mice 是 X Server 所使用的鼠标接口。基本上，Linux 的精髓就是要将所有操作系统中的信息全部都变成文件，以方便管理。

如图 3-3 所示，【/dev】下的文件，其实都是一些让用户对应到 kernel 的虚拟文件，但又介于系统模块之间，所以模块是 Ring2，而 devfs 的位置属于 Ring 1 的部分，让用户或应用程序，可以通过一些模块，使用到它提供的设备文件（device file），再经过设备文件进入 Ring 0 的 kernel 部分，这一整串就是虚拟文件系统的标准流程。

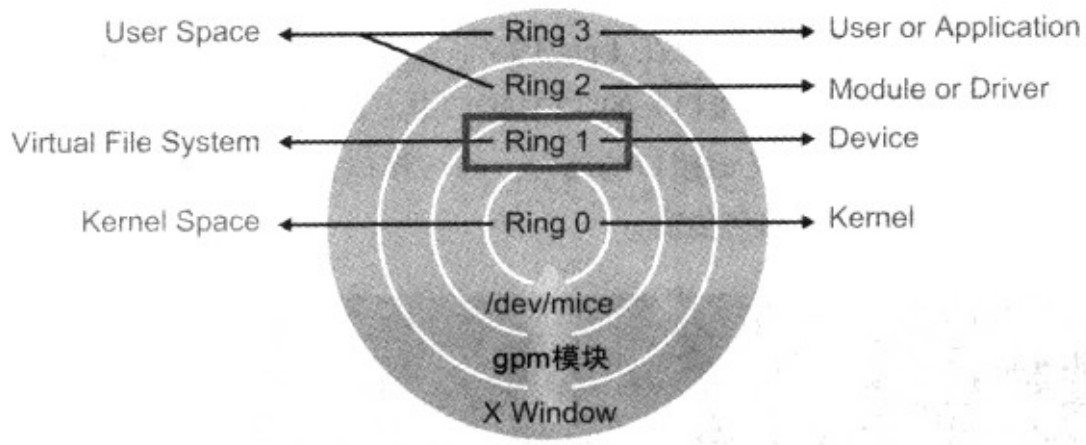


图 3-3：devfs 与操作系统的关系

如果这样的解释还是无法理解，请读者回想一下，很多 Linux 下的设备文件，都是要先经过加载模块的这一段过程，才可以看到这些设备文件的，否则这些硬件例如 U 盘（如图 3-4 所示，在加载 USB 的模块后才得以使用该设备文件），虽然可以被 Linux 所识别出来，但因为没有合适的模块让用户使用，用户将无法实际使用该硬件设备，所以在使用硬件上，模块对用户来说是非常重要的窗口。

图 3-4：USB 设备经过模块加载后才可使用

但因为 devfs 的目录架构太庞大、太凌乱，若光靠 devfs 也没有办法针对每个临时检测到的硬件做出有效的反应机制（像 U 盘插入后自动进行扫毒），对一般用户而言，也完全无法分辨哪些是可以用，哪些是未被加载的（有些 devfs 下的设备文件是永远存在的，不会因为模块的加载与否而有所差异）。所以 devfs 的缺点，就是对硬件没有一套管理的机制，所谓“管理”指的是通过硬件的检测做出适当的操作。

因此，devfs 的机制或使用方式，已经渐渐被另外一种作法——【udev】所取代，可参考第 6.1.3 节中的/etc/udev，或者是笔者的前一本著作《Linux 操作系统之奥秘》中“第 4.3.2 节：devfs vs. udev”中有针对 udev 机制的清楚解释。

对/dev 下的设备文件而言，很多用户会使用到，但不清楚其对硬件的意义在哪，图 3-5 中

是一个简单的范例，在这个例子中，可以看到当用户在【pts/1】（一般主机登录是使用/dev 下的 tty 接口，而远程或 X Window 中则是使用 pts 接口）的 TTY 接口中，这个/dev 下的接口，主要是负责做中间传输的操作（所以必须要先定义该接口的属性，如该接口是“区块”或是“字符”接口之设备文件，这样系统才知道信息该如何传输）。

```
[root@LinuxTree ~]# ps
  PID TTY          TIME CMD
 3625 pts/1    00:00:00 su
 3626 pts/1    00:00:00 bash
 3664 pts/1    00:00:00 ps
[root@LinuxTree ~]# echo "I am here" > /dev/pts/1
I am here
[root@LinuxTree ~]# echo "I am here" > /dev/pts/0
[root@LinuxTree ~]#
```

图 3-5：/dev 下设备文件的简单示意范例

当用户将一段信息传送给“pts/1”的设备文件时，系统并不将该段信息（I am here）写入【/dev/pts/1】文件中，而是将该设备文件通过字符串的方式（因该接口属性为字符串）显示在屏幕上。

只要是系统上的硬件，都会在/dev 下建立一个相应的设备文件去做对应硬件的操作，只不过有些文件会被分门别类地放在固定的目录中，像刚刚的 pts 界面正是如此，所以接下来的章节都是其中分类的结果。

如果读者想知道目前 Linux 支持哪些设备文件，可以访问以下 Linux 设备文件的官方网址，会有最详细的资料。

http://www.lanana.org/docs/device-list/

3.1.1　基本的设备文件

在/dev 目录下，有一些是基本的设备文件，却并没有被归类在子目录中，在此将一些经常用到的设备文件整理成如表 3-1 的格式，以方便和读者分享，但请注意，笔者是以 Major Number 为主要的排列顺序，并不以文件名称为主（读者可以从中发现，Major Number 其实就是一个主要的分类代码，而 Minor Number 则是其子项目），针对里面的内容若需要较完整的信息，读者可以自行在 kernel 文件的【Documentation/devices.txt】文件中找到（不过，解释当然没本书详细啰）。

表 3-1：基本的设备文件

Major	Minor	类别	设备文件名称	主要用途
1	1	char	mem	读写物理内存时用
1	2	char	kmem	读写 kernel 的虚拟内存时用
1	3	char	null	空的设备文件，意思是什么都没有，有时可以当作是接地线，把所有的信息都输出到该设备文件，所有的信息就不会出现 例：cat /var/log/message > /dev/null
1	5	char	zero	空的位文件，实际在使用上，可以用来帮用户产生一个固定大小的文件，当然文件的内容是毫无意义的 例：dd i f=/dev/zero of=/tmp/tes t.f i le bs=512 count=1
1	8	char	random	随机号码产生器
1	9	char	urandom	较快速及低安全性的随机号码产生器
1	0	block	ram0	第一个 RAM disk，一般都是用这一个设备文件当作为 initrd 的 RAM disk。当然在进入系统后就已经释放出来，所以在使用上，一样可以用作系统的 RAM disk 使用 例：1.mke2fs /dev/ram0 2.mount /dev/ram0 /mnt
2	0	block	fd0	Floppy 磁盘驱动器的设备文件
3	0	block	hda	IDE 硬盘或光驱的设备文件，这是 IDE Primary 插槽所在的 Master 接口，也就是 IDE 的第一块硬盘所使用的文件，另外也代表整块硬盘，若是第一个分区，就会使用 hda1 的设备文件，依此类推
3	64	block	hdb	请注意 hdb 设备文件的 Minor Number，这是 Primary IDE 的 Slave 界面，其 Minor Number 为 64，代表 Master（hda）最多只可以有 63 个分区，因为一个分区就会占用掉一个 Minor Number，所以分区的数目就是被 Minor Number 所限制的，这也是为何在此会将 hdb 列出来
4	0	char	tty0	TTY 的设备文件，TTY 就是用户登录时所使用的 Terminal（控制台）接口，tty0 代表第一个 Terminal，也就是用户用【Alt+F1】所进入的 Terminal，最多可以有 64 个（到 tty63）TTY

续表

Major	Minor	类别	设备文件名称	主要用途
4	64	char	ttyS0	通过 Serial Port（COM port）所使用的 Terminal 接口，最多可以使用到 192 个 serial port 界面（到 ttyS191）
5	0	char	tty	目前正在使用中的 tty 接口，也就是说，不论用户处在哪一个 tty（0,1 或 6），只要将信息送到/dev/tty 接口文件，就会直接由用户当前的 tty 界面发送
5	1	char	console	系统的终端接口，一开始启动时系统不会使用 tty，因为 tty 是给用户使用的，用户所看到的启动信息，全部都通过/dev/console 发送。不过，如果用户将信息送给/dev/console，因为是系统信息，所以一样会从用户当前的 tty 界面看到画面
5	2	char	ptmx	这是主要的控制台接口产生工具，也就是/dev/pts 目录下文件的来源，有了这一个文件，才可以让远程用户通过终端接口有登录系统的界面
7	0	char	vcs	vcs 用来对应当前所使用 virtual console（像 tty 或 pts）的文字内容，这好像很难懂；举例来说，若用户在 tty2 登录，管理者可以通过读取/dev/vcs2 文件（例如用一般的读取命令操作："cat /dev/vcs2"），就可以直接看到 tty2 当前的画面。不只是读取，也可以将信息通过不同的 vcs 设备文件，直接传送到其所对应的 TTY 等接口 例：1.echo "hello" > /dev/vcs2 2.请将窗口切换到 tty2，会在最上方看到刚才传送的 hello 信息
7	128	char	vcsa	vcsa 和上面的 vcs 设备文件的作用一样，都是用来显示目前的 virtual console 画面的，但 vcsa 又多了该 virtual console 中的属性、前置位（4 Bytes）和鼠标的位置
8	0	block	sda	SCSI 硬盘所使用的设备文件，sda 所代表的是第一块硬盘，也是完整的硬盘（也就是与分区无关），若该硬盘的第一个分区，则是 sda1，其 Minor Number 也变为 1，依此类推

续表

Major	Minor	类别	设备文件名称	主要用途
8	16	block	sdb	第二块 SCSI 硬盘所使用的设备文件，其 Minor Number 为 16，所以可以知道一块 SCSI 硬盘，最多只能有 15 个分区，因为 Minor Number 的数目只能如此分配
11	0	block	scd0	SCSI 光驱设备文件（现在通常用的都是 USB 光驱），scd0 代表第一台光驱，依此类推。有些操作系统会使用 sr 为 SCSI 光驱设备文件名称的代号，但正式文件已注明被否决掉，所以 scd 为正式 SCSI 光驱的设备文件代号
22	0	block	hdc	规则和之前的 hda 与 hdb 一样，这主要是 IDE 的 Secondary 插槽上连接的第一块（Master）硬盘，相关内容可直接参考之前 Major Number 3 的部分 至于 Third 和 Fourth 插槽上的硬盘则另外归属在 Major Number 33 和 34 中，命名规则都是一样的

3.1.2 /dev/bus

在/dev 目录中，可以根据不同的总线（bus）将硬件分类，但目前大家常用的总线访问的存储器，大概都是 USB 设备，所以在计算机中看到的应该都只有 USB 的总线在其目录中。

不过，前面就曾提到，设备文件是在 Ring1，所以用户或应用程序（Ring 3）若要使用/dev/bus 下的文件，必须要先有对应的模块（不论是外挂或 kernel 内嵌），才可以在 Linux 下使用该设备文件，当然在/dev/bus 目录下所看到的 USB 设备多少，就要先看模块所看到的数目（如图 3-6 所示）。

因为 USB 模块（uhci_hcd）所识别的 USB 控制器只有一个（usb1），所以在【/dev/bus】下的 USB 设备文件，自然也只能显示出一个（001），如图 3-7 所示。但这句话有另一个重点，若该模块所识别的 USB 信息有错误，当用户检查或使用该硬件设备时，当然问题也就跟着来，因此，模块与控制器之间的关系是很重要的，正确地说，是模块与控制器中的固件（Firmware）的关系。

```
[root@LinuxTree ~]# cat /proc/interrupts
           CPU0
  0:        236   IO-APIC-edge      timer
  1:       1195   IO-APIC-edge      i8042
  6:          4   IO-APIC-edge      floppy
  7:          0   IO-APIC-edge      parport0
  8:          1   IO-APIC-edge      rtc
  9:          0   IO-APIC-fasteoi   acpi
 12:        408   IO-APIC-edge      i8042
 14:      40981   IO-APIC-edge      libata
 15:     107726   IO-APIC-edge      libata
 16:       1223   IO-APIC-fasteoi   uhci_hcd:usb1
 17:         44   IO-APIC-fasteoi   ioc0
 18:     370815   IO-APIC-fasteoi   eth0
NMI:          0
LOC:    6015592
ERR:          0
MIS:          0
[root@LinuxTree ~]#
```

可以看到系统中USB模块认出的usb1设备

图 3-6：通过 Linux 默认的 USB 模块所看到的 USB 控制器

```
[root@LinuxTree ~]# find /dev/bus/
/dev/bus/
/dev/bus/usb
/dev/bus/usb/001
/dev/bus/usb/001/001
[root@LinuxTree ~]#
```

目录中唯一的一个文件

图 3-7：/dev/bus 下只有 USB 的设备清单

3.1.3　/dev/disk

Disk 目录下存放所有连接到这台主机上的硬盘信息，按不同的分类方式供用户查询。数据其实就是那么多，差别只在于用哪一种关键索引的方式，找出用户所要看的硬件信息，后面还会有很多不同的目录类型，都有类似的分类方式，尤其是在/sys 目录中特别明显。

图 3-8 是将/dev/disk 中的【by-label】目录做一个范例，由此可知。

- 分类方式有以下几种：【by-id】、【by-uuid】、【by-label】、【by-path】。
- 里面的数据会随着系统现有的硬件而变动（如 USB 设备）。
- 按不同的目录，应该都可以找到相同的硬件。

```
[root@LinuxTree ~]# ls /dev/disk/
by-id  by-label  by-path  by-uuid
[root@LinuxTree ~]# find /dev/disk/by-label/
/dev/disk/by-label/
/dev/disk/by-label/JUERGEN
/dev/disk/by-label/boot
[root@LinuxTree ~]# cat /proc/partitions
major minor  #blocks  name

   8     0   10485760 sda
   8     1     104391 sda1
   8     2   10377990 sda2
 253     0    9273344 dm-0
 253     1    1048576 dm-1
   8    16     503808 sdb
   8    17     503792 sdb1
[root@LinuxTree ~]# e2label /dev/sda1
/boot
[root@LinuxTree ~]# dosfslabel /dev/sdb1
Juergen
[root@LinuxTree ~]#
```

除sda1与sdb 1外，其他都是一些延伸分区或LVM机制的分区，因此都没有Label

sdb 1是U盘上的分区

图 3-8：/dev/disk 目录的分类方式

3.1.4 /dev/input

顾名思义，这是存放输入设备的目录，但主要是针对键盘和鼠标的部分，另外就是像绘图用的手写板也会在此目录中，但一般常用到的就是键盘或鼠标文件（统一的文件为 event，但鼠标可用通用的设备文件——“/dev/input/mice”）。

以文字接口下操作鼠标为例（如图 3-9 所示，用户此时位于 Ring 3 的位置），当用户在文字接口下移动鼠标时，可通过以下几个步骤控制。

Step 1 Shell 本身并不能识别鼠标，这时鼠标是没有功用的。

Step 2 必须启动“gpm”daemon（服务程序）代为加载鼠标相关的模块（如图 3-10 所示）。

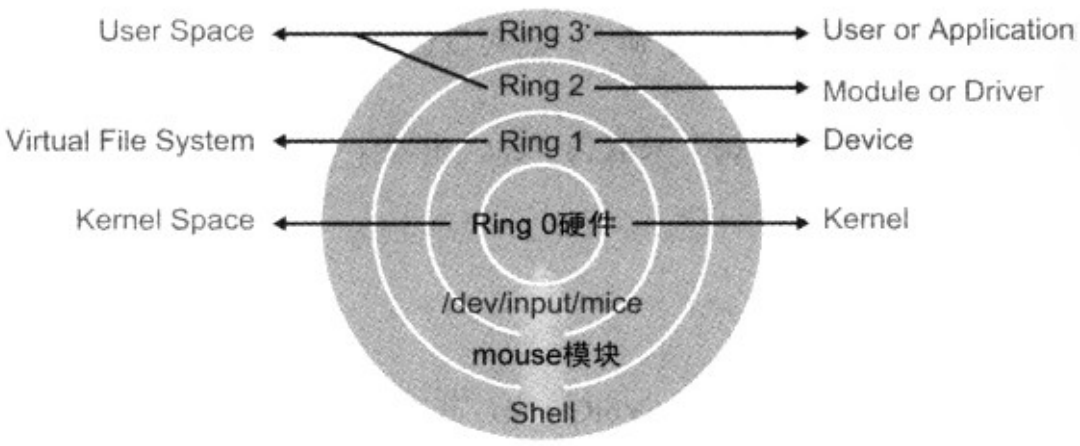


图 3-9：用户与 input 中文件的关系

Step 3　如图 3-11 所示，gpm 会自动以 sermouse 为默认的驱动程序（不称为模块是因为直接内嵌在 kernel 中的），而/dev/input/mice 则是默认的硬件设备文件。

Step 4　此时用户在 Shell 中移动鼠标，Shell 就可以通过刚刚 gpm 所建立的桥梁（mouse 所需的模块与设备文件），让鼠标能响应用户的操作。

```
    if [ -n "$IMOUSETYPE" ]; then
        if [ "$(pidofproc inputattach)" = "" ]; then
            modprobe sermouse > /dev/null 2>&1
            /usr/sbin/inputattach -$IMOUSETYPE $DEVICE --daemon
            DEVICE="/dev/input/mice"
            MOUSETYPE="exps2"
        fi
    fi
```

图 3-10：gpm 中所使用的鼠标模块

```
[root@LinuxTree ~]# ps ax|grep gpm
10993 ?        Ss     0:00 gpm -m /dev/input/mice -t exps2
11004 pts/0    R+     0:00 grep gpm
[root@LinuxTree ~]#
```

图 3-11：gpm daemon 默认所使用的鼠标设备文件

若读者想知道鼠标如何传递信息，必须先知道 mice 文件的作用就是鼠标在移动时传递数据给系统，其实只要监控该文件的内容，就等于是在监控用户传递给鼠标的指示。在此无法示范，请读者自行用以下的步骤试试看，因为这是一个动态的信息（笔者无法将截图放在书中 ☺）。

Step 1　请直接打开文件“cat /dev/input/mice”（鼠标停住不要动即可）。

Step 2　试着移动您的鼠标，这时应该就可以看到 mice 文件的变化。

Step 3　若要离开请按【Ctrl+C】即可。

Step 4　如果文字都被变成乱码，请输入【reset】命令，就可以恢复了。

Step 5　若想要看到源代码，可以使用“xxd”来取代“cat”指令，就可以了解详细的信息。

3.1.5　/dev/mapper

在“第 3.1.10 节：dev/VolGroup00”中所提到的 VolGroup，其实就是在使用 LVM 后，对系统而言的虚拟磁盘，目前的 2.6 kernel，都是 Device Mapper 实际运行产生的，先看图 3-12 针对 device mapper 和一般硬盘的对应方式。

从图 3-12 中可以知道，以往的硬盘是一个存储器的概念，在 Device Mapper 的做法下，完全被推翻，硬盘的多少不再是一个问题，随时可以新增或删除，系统只须正确通过/dev/mapper 目录下的文件，判断相应的硬盘信息即可，因此在 LVM 的机制下，/dev/mapper 目录的重要性自然是不言而喻的。

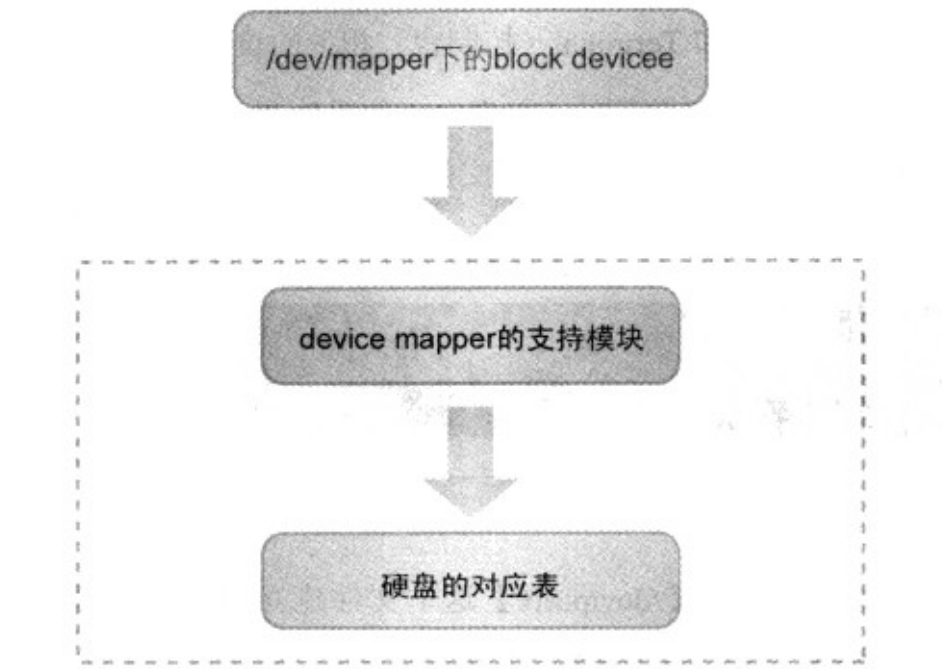


图 3-12：Device Mapper 示意图

在/dev/mapper 目录中有哪些文件？从图 3-13 中可看到就是当初/dev/VolGroup00（视 Group 数量而定）目录中所链接到的文件，所以，/dev/mapper 才是真正会使用到的设备文件。

/dev/mapper 中还有一个 control 文件，它是 Device Mapper 的控制文件，所以一定需要，不过，因为是 devfs 自动产生的，即使启动后不小心删除，重启后一样会自动还原。

```
[root@LinuxTree ~]# ls /dev/mapper/
control  VolGroup00-LogVol00  VolGroup00-LogVol01
[root@LinuxTree ~]# ls /dev/VolGroup00/
LogVol00  LogVol01
[root@LinuxTree ~]#
```

图 3-13：/dev/mapper 目录下的文件

3.1.6 /dev/net

目录中默认只会有一个【tun】设备文件，因为此文件是用来建立 VPN 的“tunnel”所使用

的，如果没有这个文件，VPN 就没有交互管道。

3.1.7　/dev/pts

pts 的全名为 Pseudo-terminal slave，另一个和 pts 有直接关系的是 ptmx，全名为 Pseudo-terminal master。由这两个名称可以知道它们都是和 Terminal（控制台）相关的设备文件。

/dev/pts 目录下的文件，都是当用户通过非本机登录时，产生出可使用的 Terminal 界面（如图 3-14 所示），像“0”或“1”这两个文件。这些虽然都只是文件名称，但可表示目前有几个通过远程或 X Window 的用户在在线操作。

```
[root@LinuxTree pts]# pwd
/dev/pts
[root@LinuxTree pts]# ls
0  1
[root@LinuxTree pts]#
```

图 3-14：/dev/pts 下的文件

其实【/dev/pts】目录下的所有文件，都是由【/dev/ptmx】这个文件建立的，也就是说，ptmx 实际为 Terminal 的 Master，而 pts 则是 Terminal Slave。

由图 3-15 可以明显看出，在同一时间（差一秒是因为笔者手太慢啰），底层的 Terminal 中，用 cat 命令对【/dev/ptmx】做读取的操作，造成在上层的 Terminal 中，【/dev/pts】目录新产生了一个【3】的文件，如果 cat 命令是一个用户 A，这个“3”文件，就是被 ptmx 所派来服务用户 A 的 Terminal 接口。

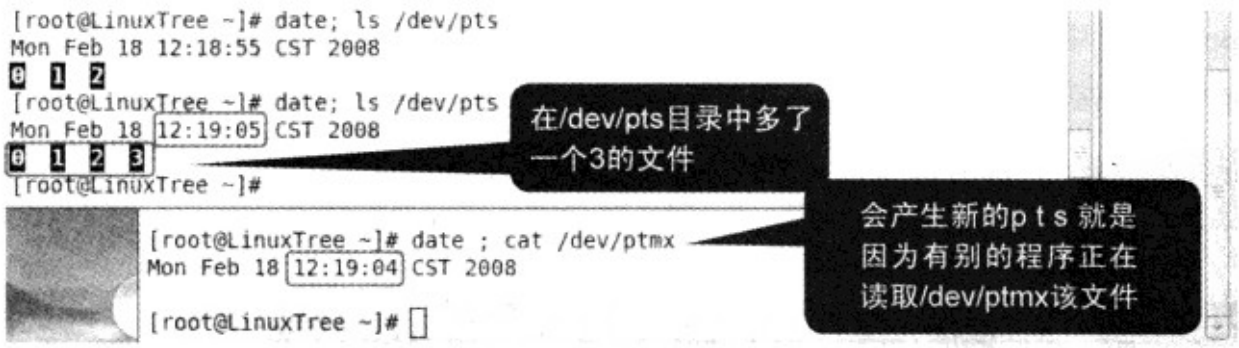


图 3-15：ptmx 与 pts 的关系

通过以上范例可知，/dev/pts 目录的主要目的在于存放【ptmx】为用户所产生出的 Slave Terminal 接口的设备文件，其使用的场合为 X Window 下的 Terminal 窗口（xterm 软件），以及 ssh 登录时的接口，所以其实是很常用的。

3.1.8 /dev/shm

这是一个很特别的目录，shm 是 Linux 专门用来分享内存的一种 API，也就是说，可以通过 shm 的机制，方便地让内存直接变为一种可让用户读写数据的使用空间。

如图 3-16 所示，启动时系统默认会将/dev/shm 的目录以 tmpfs 的文件系统格式（在之前的 Linux 版本，可能会称为 shmfs，但目前都称其为 tmpfs）挂载，这是一个很特别的操作，因为一般 tmpfs 的目录是不会选择放在“/dev”目录下的。

```
[root@LinuxTree ~]# mount | grep shm
tmpfs on /dev/shm type tmpfs (rw)
[root@LinuxTree ~]#
```

图 3-16：/dev/shm 的文件系统格式

这也说明了，tmpfs、proc 和 sys 一样，都是属于虚拟的文件系统，因为不需要任何的实际储存位置，因此，启动或 rescue mode 都大量使用虚拟文件系统，以建立及储存相关的信息。tmpfs 对用内存当硬盘使用的用户来说十分方便，甚至在 Linux 系统进入后，其实也已经默认建立好一个 tmpfs 的目录以供用户使用，但可使用的大小受限于内存大小，因为 tmpfs 所使用的空间来自物理内存。

tmpfs 简单好用，只要以 tmpfs 的格式将要用的目录 mount 起来，就可以直接指向内存的资源，重要的是读写速度比一般的硬盘快多了，若用户在某些情况下需要高速读写时，可以考虑采用 tmpfs 的方式建立一个临时的空间，毕竟，有哪一种硬盘的读写速度大于内存的读写速度？

通过以下简单示例来介绍如何在 Linux 下建立一个 tmpfs 的文件系统。

Step 1 建立一个目录/mnt/tmpfs（如图 3-17 所示）。

```
[root@localhost ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             6.7G  1.7G  4.7G  26% /
tmpfs                 252M     0  252M   0% /dev/shm
[root@localhost ~]# mkdir /tmp/tmpfs
[root@localhost ~]# ls -ld /tmp/tmpfs/
drwxr-xr-x 2 root root 4096 May 25 20:29 /tmp/tmpfs/
```

图 3-17：建立一个目录

Step 2 以 tmpfs 的文件系统格式 mount 到/mnt/tmpfs（如图 3-18），【-o size=50M】设定使用的内存大小为 50 MB，若没做此设定，默认会将所有目前剩余的内存全部允许/mnt/tmpfs 使用，这样当目录中的文件大小超过内存剩余空间时，就会产生内存空间不足的问题，就算没超过，也容易因占用太多的内存空间而导致系统过慢，因此，建议还是先行手动设定使用大小，以免出现问题。

```
[root@localhost ~]# mount -t tmpfs -o size=50M tmpfs /mnt/tmpfs/
```

图 3-18：以 tmpfs 的文件系统格式 mount 到/mnt/tmpfs

Step 3 以 df 命令检查设置是否正确（如图 3-19 所示），mount 成功后，该目录的大小应为设定的大小（50MB）。

```
[root@localhost ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/sda1             6.7G  1.7G  4.7G  26% /
tmpfs                 252M     0  252M   0% /dev/shm
tmpfs                  50M     0   50M   0% /mnt/tmpfs
[root@localhost ~]# _
```

图 3-19：以 df 命令检查设置是否正确

Step 4 /mnt/tmpfs 已经成为一个 tmpfs 文件系统的目录。

如一开始所说，在 Fedora 启动时，便已经将/dev/shm 目录以 tmpfs 的方式 mount 起来了，但请注意，因为/dev/shm 是以不限大小的方式 mount 的，所以，其可用空间将会是全部可用的内存大小，若不注意使用，内存将使用过量并造成系统问题。

3.1.9　/dev/.udev

在 udev 的机制中，有大量的数据都是来自/dev 目录，因为其一开始的目的就是为了改善与取代 devfs，也就是/dev 目录下的杂乱存放方式，因此，在这个目录中，udev 也放了一些相关信息的文件在其中，里面记录的都是启动时经过 udev 整理出来后，成功（变为记录之一）或是出现问题的文件。

udev 在存放时便以其在/sys 目录中（sysfs 所产生出的目录内容太多，详情请参考《Linux 操作系统之奥秘》一书）的位置为文件名称，但请注意，在【db】目录中的这些文件大部分都是一些链接文件，因为这些成功的文件，也都是经由 udev 的 rules（/etc/udev/rules.d）所建立的。

以图 3-20 为例，在【db】目录中的第一笔数据其文件名为“class@vc@vcsa7”，请将“@”视为一个“目录”的分隔识别符号，也就是说，如果在该目录中找得到，就一定可以找得到“/sys/class/vc/vcsa7”这个目录。

```
[root@LinuxTree .udev]# ls
db  failed  uevent_seqnum
[root@LinuxTree .udev]# find db/!head -5
db/
db/class@vc@vcsa7
db/class@vc@vcs7
db/class@vc@vcsa5
db/class@vc@vcs5
[root@LinuxTree .udev]# ls /sys/class/vc/vcsa7/
dev  power  subsystem  uevent
[root@LinuxTree .udev]#
```

图 3-20：/dev/.udev 目录中的部分信息

不过，这些链接文件是无法在/dev/.udev/下直接查看的，除非是非链接文件的部分，才可以显示其内容，也就是该硬件的信息，如【/dev/.udev/db /block@ram0】。

3.1.10 /dev/VolGroup00

如果计算机在安装时（或是日后使用到）使用 LVM 的方式处理硬盘，就会在/dev 中有这个目录，但 VolGroup 后面所带的编号，则要视用户使用多少个 LVM 的 Group 而定（可多块磁盘整合为一个 VolGroup），不过，里面的文件只是链接到/dev/mapper 目录中的 LVM 的设备文件（如图 3-21 所示），详细内容请参考“第 3.1.5 节：/dev/mapper”的介绍。

```
[root@LinuxTree db]# ls -l /dev/VolGroup00/
total 0
lrwxrwxrwx 1 root root 31 2008-02-18 09:54 LogVol00 -> /dev/mapper/VolGroup00-Lo
gVol00
lrwxrwxrwx 1 root root 31 2008-02-18 09:54 LogVol01 -> /dev/mapper/VolGroup00-Lo
gVol01
[root@LinuxTree db]#
```

图 3-21：/dev/VolGroup00 中的连接信息

3.2 程序信息与系统设置目录【/proc】

proc 目录是由 procfs 的文件系统所产生出来的，procfs 是 Linux 用以储存所有开机后硬件、process 相关信息的方式，全名为 process file system，主要目录在【/proc】下，但【/proc】并不是一个实体的目录，即使是 root 也无法在其中增加或删除文件。此目录其实是 kernel 加载后，在内存里面建立的一个虚拟目录，有专属的文件系统格式 procfs，主要提供系统一些实时的信息，因为是实时的，所以文件及目录都会随时变动。要注意的是，procfs 因为是建立在内存中属于 kernel 层的文件，Linux 为了保障系统的稳定度，不论是在 initrd、rescuemode 或是 Linux 主要系统，都无法对【/proc】下的文件做读写的操作，但有些文件是属于开关性质，可以通过这

些文件做实时变更属性的操作。

procfs 既然是虚拟文件系统，当然也就没有限定要在哪一个目录中才看得到其内容，也就是说，【/proc】只是一个默认的目录，可以用任何一个用户所希望使用的目录名称来使用 procfs，像“/procinfo”（如图 3-22 所示）。

图 3-22：在新建立的/procinfo 目录中使用 procfs

/proc 目录里面的数据都是根据目前系统的状态所产生的，当作系统核心架构的接口文件，所以绝大部分数据都只能供用户做读取的操作，而不能任由用户修改；某些文件可以变更，是因为使用核心变量的方式，例如在做 NAT 时最常使用到的“/proc/sys/net/ipv4/ip_forward”文件就是一个例子。其实/proc 这个目录在【proc man page】的使用手册中，已经将所有目录及文件都做了简单的介绍，读者有疑问时，也可以参考 Linux 下对 proc 的介绍。

/proc 对系统而言，主要有几种功用：

- 整理系统内部的信息。
- 存放主机硬件信息。
- 调整系统执行时的参数。
- 检查及修改网络和主机的参数。
- 检查及调整系统的内存和性能。

3.2.1 基本程序文件

因为/proc 目录目前有很多重要的信息在其中，光是/proc 下的文件就有许多很值得参考，所以在进入目录之前，先介绍几个笔者认为一定要知道或很重要的文件给各位读者，或许下次在使用 Linux 时，可以看一下下面这几个文件。

■ cpuinfo

这个文件有非常重要的信息，因为该文件代表 CPU 的硬件信息（如图 3-23 所示），任何命令或文件，只要是和 CPU 有关的，都会参考这个文件。也就是说，如果 BIOS 或 CPU 有问题，所有相关的 CPU 功能都会一并出错，也就无法使用该文件，因为它显示的就是用户可使用的部分。

若用户将 Intel 的 SpeedStep 或 AMD 的 Cool and Quiet 功能（也就是一般常听到的 CPU 省电模式）打开，将可以发现在 cpu MHz 的部分会随 CPU 的频率变化动态调整，当然，若变动的方式有问题，就代表此功能也出了问题。

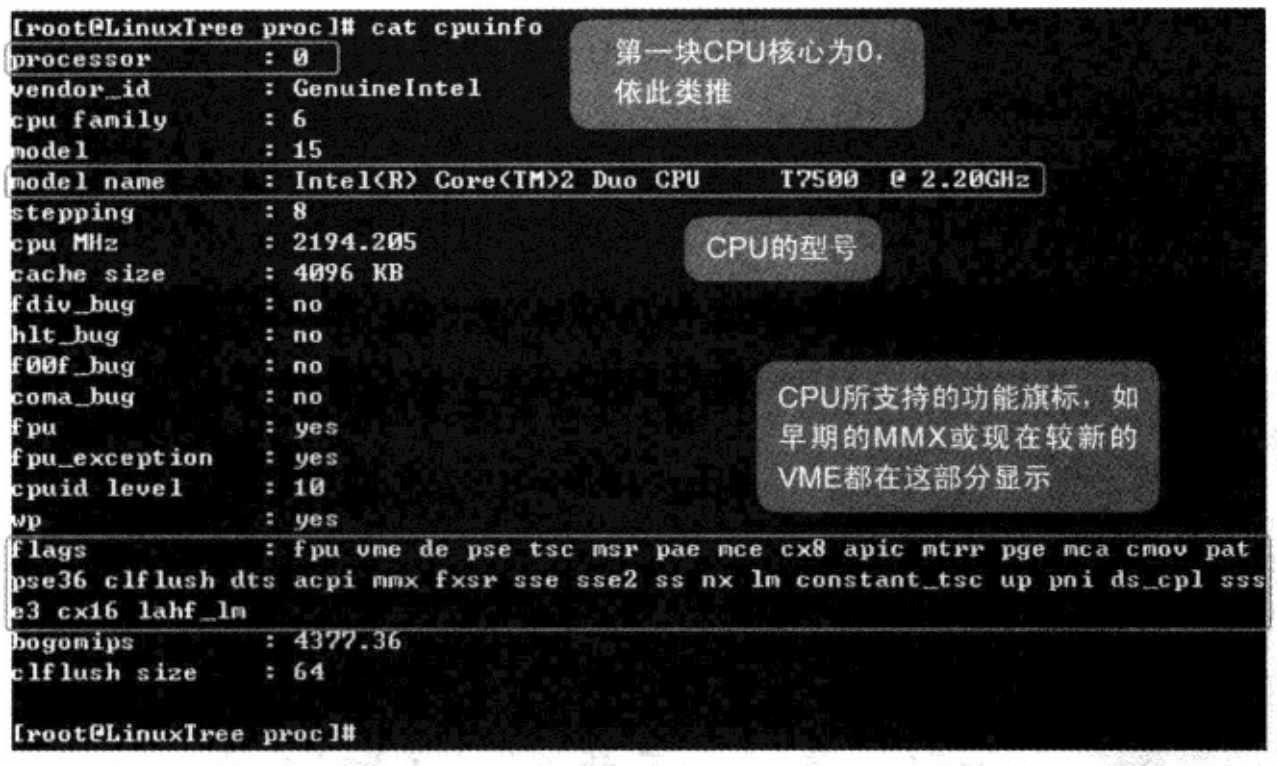


图 3-23：/proc 中 CPU 的硬件信息

■ devices

里面记录所有在/dev 目录中相关的设备文件分类方式，最基本的分类方式就是以 Major Number 为标准（如图 3-24 所示，红色部分即为 Major Number 的编号），由该编号再用 Minor

```
[root@LinuxTree proc]# head -10 devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  6 lp
  7 vcs
[root@LinuxTree proc]#
```

图 3-24：devices 文件中的部分信息

Number 及设备文件种类细分。

■ interrupts

它主要让用户可以查看每一个 IRQ 的编号对应到哪一个硬件设备（见图 3-25），具体分为以下 4 个字段说明：

① IRQ 编号，这和 BIOS 中的定义是一致的，若用户在 BIOS 中做调整，这里也会跟着变换。

② interrupts 的使用次数，笔者的模拟环境是单核的 CPU，若用户遇到多核的 CPU 时，字段会自动按核心数目调整增加。

③ Interrupt 的管理模式，一般都是使用 Intel 所提供的 APIC（Advanced Programmable Interrupt Controllers）架构，但这只在支持 SMP（Symmetric Multi-Porcessor）的计算机中才可以使用，若不支持就可能会改为 PIC（Advanced Programmable Interrupt Controllers）架构。

```
[root@LinuxTree ~]# cat /proc/interrupts
           CPU0
  0:        237   IO-APIC-edge      timer
  1:       4926   IO-APIC-edge      i8042
  6:          4   IO-APIC-edge      floppy
  7:          0   IO-APIC-edge      parport0
  8:          2   IO-APIC-edge      rtc
  9:          0   IO-APIC-fasteoi   acpi
 12:       3693   IO-APIC-edge      i8042
 14:      25578   IO-APIC-edge      libata
 15:      48273   IO-APIC-edge      libata
 16:          0   IO-APIC-fasteoi   uhci_hcd:usb1
 17:         44   IO-APIC-fasteoi   ioc0
 18:      41613   IO-APIC-fasteoi   eth0
NMI:          0
LOC:    2331036
ERR:          0
MIS:          0
[root@LinuxTree ~]#
```

图 3-25：interrupts 文件中的记录

④ 对应到的硬件设备，在此如果发现对应上有问题（像打印机所使用的 partport0，若对应到不正确的 IRQ 或有相冲突的情况），就有可能导致该设备无法使用，这不是不可能发生的问题。

■ ioports

这个文件将目前系统上所有“可看到的硬件对应到内存位置的分配表”的详细信息呈现出来（如图 3-26 所示），但请注意，该文件较前面的部分（A 部分），是一些 kernel 默认就支持的硬件，其对应的地址是单一的；而下半部（B 部分）另外经过加载模块才可使用的硬件地址，则会由模块的硬件地址再往下延伸。所以这是一份非常有用的信息，因为当模块加载有问题，或是加载后使用有问题时，可以对应看一下这一份数据，或许是模块下根本就没有相应的硬件在其中（这时不是内存分配有问题，就是硬件有问题而导致无法使用）。

```
[root@LinuxTree proc]# head -5 ioports
0000-001f : dma1
0020-0021 : pic1
0040-0043 : timer0
0050-0053 : timer1
0060-006f : keyboard
[root@LinuxTree proc]# tail -5 ioports
  1060-107f : uhci_hcd
1080-10ff : 0000:00:10.0
1400-147f : 0000:00:11.0
  1400-141f : pcnet32_probe_pci
1480-148f : 0000:00:0f.0
[root@LinuxTree proc]#
```

A

B

图 3-26：ioports 文件的部分信息

■ kcore

这一个文件其实就是系统上可以检测到的物理内存，换句话说，主机上有多大的内存，这个文件就应该有多大。但是，如果在启动时用【mem=xxxM】的参数去改变内存的大小，kcore 文件的大小也会随之而变。不过要知道，实际上这个文件并没有占硬盘的空间，因为它不过是一个类似显示物理内存的状态文件，如果用【du kcore】去检查所占硬盘的空间，就会发现其值为“0”，如图 3-27 所示。

```
[root@LinuxTree proc]# ls -lh kcore
-r-------- 1 root root 513M 2008-02-22 20:22 kcore
[root@LinuxTree proc]# free -m
             total       used       free     shared    buffers     cached
Mem:           503        340        162          0         29        260
-/+ buffers/cache:         49        453
Swap:         1023          0       1023
[root@LinuxTree proc]#
```

图 3-27：kcore 和物理内存大小的比较

■ keys 和 key-users

这两个文件属于同一功能，都是属于 Linux 密钥保留服务（Linux key retention service）所使用的文件，其服务是 2.6 kernel 所提出的新方案。主要的目的是要将一些身份验证的文件暂存到 kernel 中，但并不是每一个操作系统都会用这种方式，像 SuSE 就没有采用这方案（笔者的版本为 SLES10 SP1）。

将这两个文件的字段属性分别列入下表（请与图 3-28 一起看），为避免翻译错误（笔者可不是加密或安全性的高手），这里全部以原文表示：

- keys

1	2	3	4	5
Serial	Flags	Usage	Expiry	Permissions
6	7	8	9	10
UID	GID	TypeName	Description	Summary

- key-users

A	B	C	D	E
UID	Usage count	Total number of keys and number instantiated	Key count quota	Key size quota

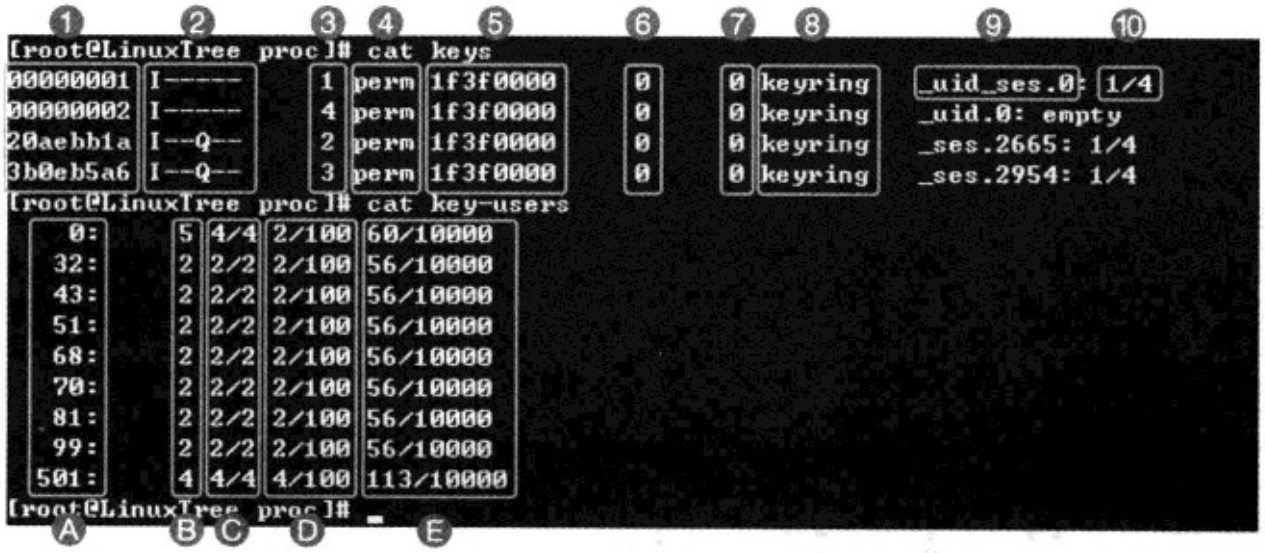


图 3-28：keys 和 key-users 文件的内容

如果读者刚好需要这一部分详细的数据或实现方式，可以参考网上 IBM 的文章，网址如下：

http://www.ibm.com/developerworks/linux/library/l-key-retention.html

■ kmsg

在系统尚未进入操作系统阶段，还在加载 kernel 及执行 initrd 时，会将信息先记录在/proc/kmsg 文件中（因为在 initrd 阶段前期，并没有实际的硬盘可供记录），等进入操作系统执行完 klogd 后，klogd 再将/proc/kmsg 的所有内容全部写入/var/log/message 文件中（如图 3-29 所示），这也是为何在/var/log/message 文件的一开始，依然可以看到刚启动在加载 kernel 阶段和 initrd 阶段的信息。

图 3-29：kmsg 开始被写入的时间点

■ meminfo

该文件中所记录的是系统内存的信息，因为 Linux 以硬件为主，所以当内存的大小读取错误时，就会显示错误，相应地，系统所能使用的内存大小，自然就跟着错误。

甚至如果出错的部分和程序有关系，执行时就会发生问题，因此，这个文件在正常的时候只是用来查看内存大小，但有时也可以用来判断是否有内存读取错误的情况，但如果需要细节信息，还是必须看下一页的 iomem 或 mtrr 这两个文件的信息，才可以做更正确的判断，如图 3-30 所示。

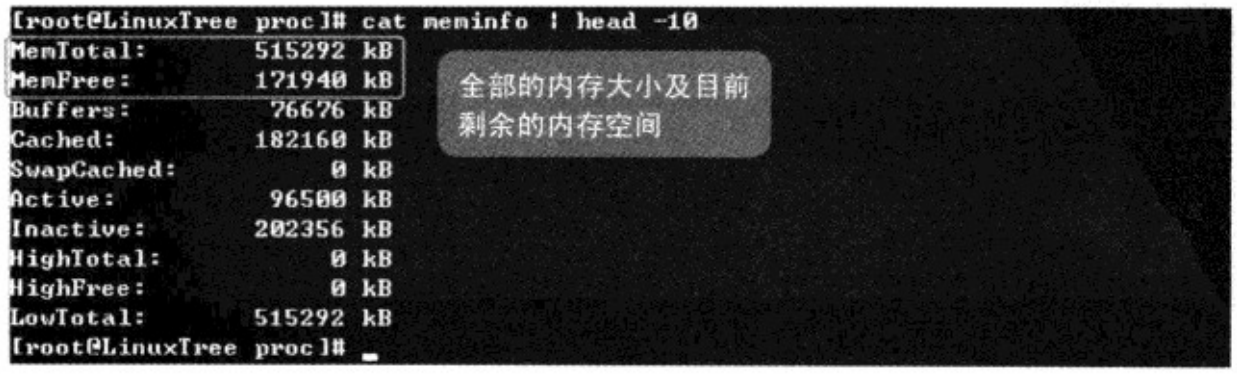


图 3-30：/proc中meminfo文件的信息

■ modules

如果各位读者用过【lsmod】命令查询目前系统上所使用的模块有哪些，这个文件就是该命令所参考的根据（如图 3-31 所示），其实【lsmod】命令不过是帮用户把/proc/modules 中比较凌乱的内容，整理成字段比较整齐的表现方式，并加个字段名称，但主要内容完全是/proc/modules 文件的内容。

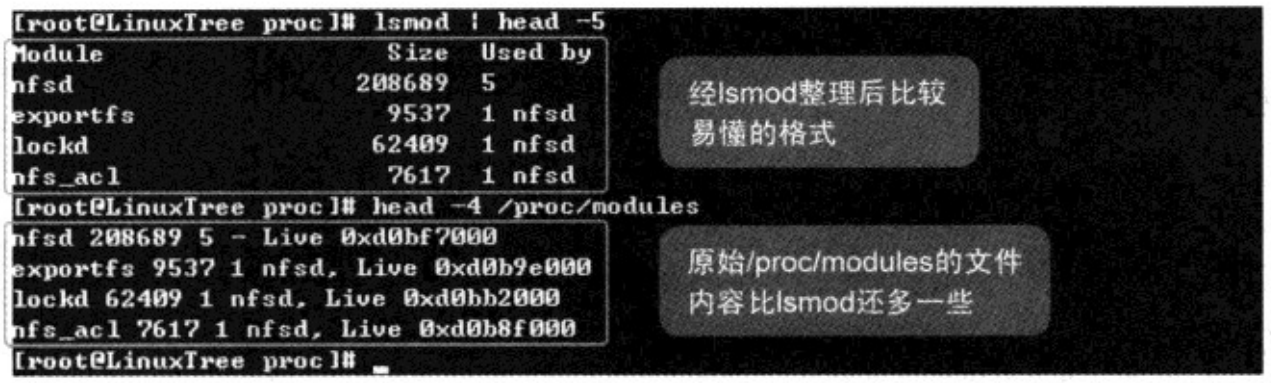


图 3-31：/proc/modules 和 lsmod 之间的关系

■ mtrr 和 iomem

观察系统中内存的配置，必须检查两个在/proc 目录下的主要配置记录文件：mtrr 及 iomem。简单地说，mtrr 是负责内存配置的机制，而 iomem 则是储存配置后所有内存储存的明细信息，如果再配合前面所提到的 ioports，则会更完整地描述内存状况。

在说明这两个文件时，会以 mtrr 为主去辅助检查 iomem 的配置情况，两者与内存的关系用图 3-32 来解释会比较易懂。主要是让用户在读写设备时，可以通过 mtrr 及 iomem 文件，找到实际存放在内存的一段空间，而这一段空间有可能是适配器的信息，或是用户的程序及 kernel 的 data。

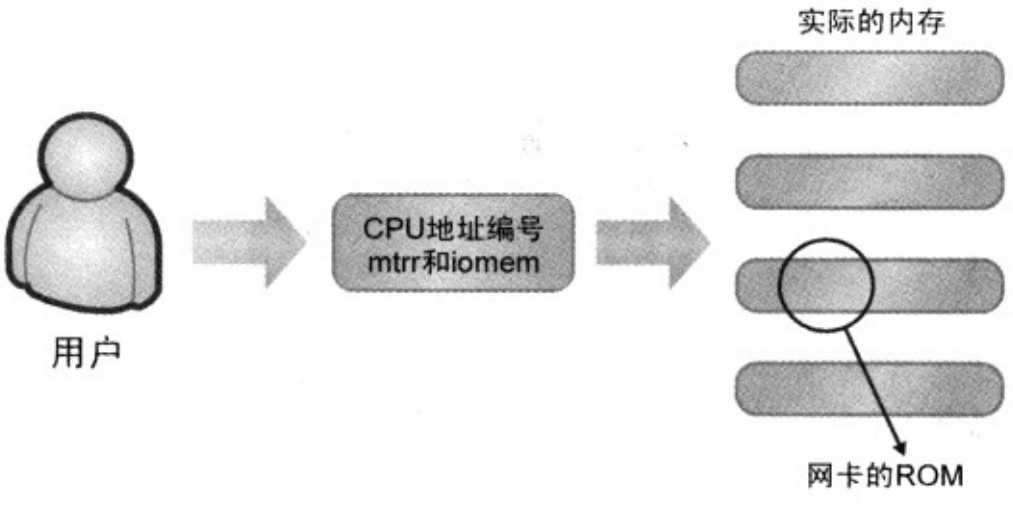


图 3-32：mtrr 与 iomem 的示意图

但 mtrr 与 iomem 并不是完全按照顺序一一对应的，mtrr 与 iomem 两个的对应关系看图 3-33 的说明就会比较清楚。要注意的是，图 3-33 中左边部分是原本 mtrr 内容，为了方便读者比对，被笔者拆成一条一条的，而右边则是 iomem 的原始内容。

图 3-33：mtrr 与 iomem 的对应图

这里只是一部分的信息，因为 mtrr 与 iomem 的东西实在太多，但从上面的解释应该不难理解其相互依存的关系，全部细节因为较偏硬件的部分，请有兴趣的读者自行参考笔者的著作《Linux 操作系统之奥秘》。请读者注意，每个系统的内存配置并不完全一样，所以当自行检查时，若出现不一样的配置结果是很正常的事。

■ partitions

在较早的 Linux 版本加入一块硬盘或是插入一个 U 盘时，将有一个很大的问题，就是不知道到底挂载到哪一个分区才对，这个文件可以实时呈现系统目前所看得到（看不到代表有问题）的分区，如图 3-34 所示。

```
[root@LinuxTree proc]# cat /proc/partitions
major minor  #blocks  name

   8     0   10485760 sda
   8     1     104391 sda1
   8     2   10377990 sda2
 253     0    9273344 dm-0
 253     1    1048576 dm-1
[root@LinuxTree proc]#
```

图 3-34：目前系统所认得的所有分区

3.2.2 /proc/[number]

一进入/proc 目录中，一定会看到非常多“数字”的目录，图 3-35 中所圈起来的那一大片都是，这些数字都是很有意义的目录，因为它们代表着所有目前正在系统中运行的所有程序（如果读者不是很了解程序，简单地说，硬盘中的软件或程序，执行后便成为程序，存在内存中），也就是各位时常使用【ps】命令所显示出的所有项目。

```
[root@LinuxTree ~]# ls /proc/
1     1856  2458  2786  3072  3157  4          fs           schedstat
1009  1925  2506  2787  3077  3159  404        interrupts   scsi
107   1964  2537  2788  308   3163  46         iomem        self
108   2     2569  2856  309   3165  47         ioports      slabinfo
111   2007  2586  2860  3092  3166  5          irq          stat
113   2036  2658  2862  3094  3193  6          kallsyms     swaps
1228  2069  2661  2890  3096  320   7          kcore        sys
1235  2080  2678  2937  3101  326   986        keys         sysrq-trigger
1238  2093  2679  3     3116  3267  acpi       key-users    sysvipc
1241  2141  2695  304   3118  3268  buddyinfo  kmsg         timer_list
1244  2162  2696  3042  3119  3269  bus        loadavg      timer_stats
1250  2186  2707  3045  3120  3305  cmdline    locks        tty
1251  2226  2708  3046  3121  3306  cpuinfo    mdstat       uptime
136   2231  2718  305   3124  331   crypto     meminfo      version
137   2248  2723  3052  3125  339   devices    misc         vmcore
138   2267  2775  3055  3128  3470  diskstats  modules      vmstat
139   2284  278   3057  3130  367   dma        mounts       zoneinfo
1718  2309  2782  3060  3137  3699  driver     mpt
1819  2317  2783  3062  3144  3701  execdomains  mtrr
1821  2334  2784  3065  3153  3703  fb         net
1853  2381  2785  3070  3155  3989  filesystems  partitions
[root@LinuxTree ~]#
```

图 3-35：/proc 下的数字目录

在这些“数字”目录中，有很多信息是可以供用户参考的。

图 3-36 是将/proc 中两个数字目录打开比较，可以发现内容是一模一样的，因为每个程序该有哪些目录及哪些文件，其实都是已经被 procfs 所定义好的，不能更改，所以所有的数字目录都是只读的。

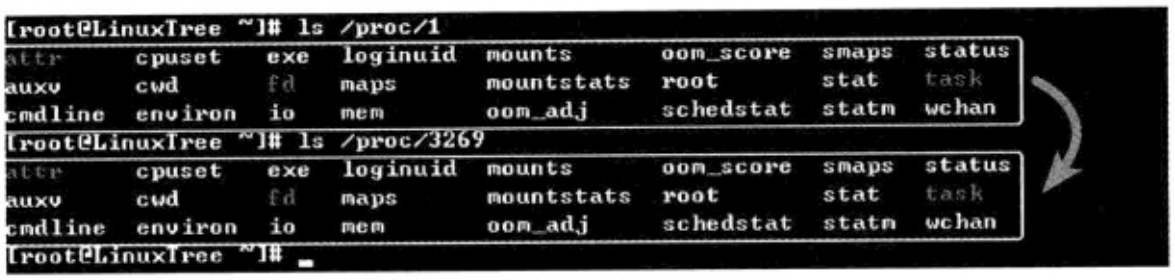


图 3-36：/proc 下数字目录中的内容

在每一个数字目录中，含有大量的子目录及文件，这里以“程序 1”的目录为例，因为这是大家都有的目录——“init”的程序。但因为文件实在太多，所以不会在此全部列出，如果不在此书中而读者有兴趣的，大部分都可以在 Linux 的【proc man page】中找到相关的信息。

- cmdline：该程序所使用的完整命令，其实就是使用【ps ax】时所看到【COMMAND】字段所显示的信息。
- cwd：该程序所使用的目录，这个文件只是一个链接文件，若要看这程序所在的目录，只需要用【ls –l /proc/1/cwd】即可。
- maps：这文件很有趣，是该程序所在的内存地址区段，除了有该程序所使用到的每一个文件清单外，还会将其访问的权限也列在其中。
- smaps：这文件的内容和 maps 其实是一样的，都在于显示出该程序的存储器使用方式，但比 maps 更为详细，因为它将每一区段的详细使用情况都记录下来了。
- stat：记录所有该程序的状态，只不过全部都是以数字代码的方式记录下来的，所以看该文件是一堆数字，如果想知道每一个数字代表的意义，一样可以在 proc 的【man page】中的 stat 找到详细的说明。
- statm：这是在记录内存的状况，和刚刚的 stat 一样，都是用数字记录下来的，所以要知道其意义还是要对照【man page】。
- status：如果对刚刚 stat 文件不满意，太难理解或信息太少，可以参考 status 这个文件，不但提供比 stat 还多的信息，甚至有字段的名称供用户看，在读取上方便许多。

3.2.3 /proc/acpi

ACPI 是 Advanced Configuration and Power Interface 的简写，自然这下面放的都是一些省电技术相关的文件，比如说，CPU 的省电支持程度、可被唤醒（wake up）的接口、电源的使用程度等都属于这范围。

不过，因为/proc 是慢慢要被 sysfs 的/sys 完全取代掉的，所以在/proc/acpi 里的信息，笔者觉得似乎少了一点，其中很多的文件及目录都是空的，感觉虽然有其机制，但好像没有完全利

用到，因为在/sys 目录下的电源相关信息完整多了。

例如 Intel 的 SpeedStep 或 AMD 的 PowerNow，以及主机电源的调整方式、CPU 的省电模式、目前的系统省电状态，等等，/sys 都比/proc/acpi 下的整合信息来得完整，所以建议读者还是去看/sys 下的信息较好。

如果读者的系统依然是在 2.4Kernel，那就还是必须要以/proc/acpi 为电源管理的主要参考路径。

3.2.4　/proc/bus

有关该主机上现有总线的所有信息，包括输入设备、PCI 接口、PCMCIA 扩充卡及 USB 都在其中，不过，在此目录中的信息并不是很完整，所以，如果要细看某一特定接口，还是回到像 lspci 或/sys 目录中去找比较清楚，如图 3-37 所示。

```
[root@LinuxTree bus]# pwd
/proc/bus
[root@LinuxTree bus]# ls
input  pccard  pci  usb
[root@LinuxTree bus]# cat input/
devices   handlers
[root@LinuxTree bus]# cat input/handlers
N: Number=0 Name=kbd
N: Number=1 Name=mousedev Minor=32
N: Number=2 Name=evdev Minor=64
[root@LinuxTree bus]# _
```

图 3-37

3.2.5　/proc/driver

在 proc 的定义中，这目录应该是空的，不过这目录好像都会有一些很零星的 driver 文件在其中，意思是说，并不是每一种硬件的 driver 信息都会在这目录中（可以说很少），所以笔者认为要用到此目录的机会真的很少。

3.2.6　/proc/fs

这也是一个定义为空目录的架构，和刚刚比较不一样的是里面真的没有一个文件。

3.2.7　/proc/irq

这个目录存放着非常多的子目录，每一个子目录的名字就是 IRQ 的编号，如 0、1、2，等等，在这些 IRQ 编号目录中，会有一些子目录，可以通过这些子目录了解目前主机上占用这些 IRQ 的设备是哪些，如图 3-38 所示。

```
[root@LinuxTree irq]# find !head -10
.
./18
./18/eth0
./18/smp_affinity
./17
./17/ioc0
./17/smp_affinity
./16
./16/uhci_hcd:usb1
./16/smp_affinity
[root@LinuxTree irq]# _
```

图 3-38：/proc/irq 下的文件名称

看到这里，要跟读者说声抱歉，/proc/irq 目录可以不用看了，因为在/proc 下有一个之前提到的文件 interrupts，直接读取这个文件的内容，既清楚又简单，不像 irq 这样的目录太过简单又不方便，如图 3-39 所示。

```
[root@LinuxTree ~]# cat /proc/interrupts
           CPU0
  0:        237   IO-APIC-edge      timer
  1:       4926   IO-APIC-edge      i8042
  6:          4   IO-APIC-edge      floppy
  7:          0   IO-APIC-edge      parport0
  8:          2   IO-APIC-edge      rtc
  9:          0   IO-APIC-fasteoi   acpi
 12:       3693   IO-APIC-edge      i8042
 14:      25578   IO-APIC-edge      libata
 15:      48273   IO-APIC-edge      libata
 16:          0   IO-APIC-fasteoi   uhci_hcd:usb1
 17:         44   IO-APIC-fasteoi   ioc0
 18:      41613   IO-APIC-fasteoi   eth0
NMI:          0
LOC:    2331036
ERR:          0
MIS:          0
[root@LinuxTree ~]# _
```

和图3-38比较起来，是不是信息完整多了，版面又比较干净

图 3-39：/proc/interrupts 文件的内容

3.2.8 /proc/net

在 net 目录中都是一些网络相关的虚拟文件，每一个文件也都是 ASCII 文件，所以都可以直接使用【cat】命令来观看其内容，不过其实用一般常用的命令，像“ifconfig”、“arp”或“netstat”等会方便得多，因为这些文件都是一些难以直接理解的文字文件。

里面很多文件都是如此，基本上，大部分网络的命令其实都是帮用户方便读取的文件，在此举出几个文件供读者参考。

■ arp

“/proc/net/arp”文件就是一般网管人员在使用【arp】命令时所看到的结果，该文件完全以最原始的结果记载，而【arp】命令则会帮助用户再做一次翻译，让用户更好地进行阅读，

如图 3-40 所示。

```
[root@LinuxTree net]# arp
Address                  HWtype  HWaddress           Flags Mask            Iface
10.32.15.254             (incomplete)                                      eth0
10.32.15.80              ether   00:18:F3:AC:21:E8   C                     eth0
10.32.15.207             ether   00:1C:23:FC:9C:6B   C                     eth0
10.32.15.1               (incomplete)                                      eth0
10.32.12.1               ether   00:00:0C:07:AC:0A   C                     eth0
[root@LinuxTree net]# cat arp
IP address       HW type     Flags       HW address            Mask     Device
10.32.15.254     0x1         0x0         00:00:00:00:00:00     *        eth0
10.32.15.80      0x1         0x2         00:18:F3:AC:21:E8     *        eth0
10.32.15.207     0x1         0x2         00:1C:23:FC:9C:6B     *        eth0
10.32.15.1       0x1         0x0         00:00:00:00:00:00     *        eth0
10.32.12.1       0x1         0x2         00:00:0C:07:AC:0A     *        eth0
[root@LinuxTree net]#
```

直接以命令方式查看arp table

读取/proc/net/arp文件的结果

图 3-40：用 arp 命令直接查看 arp 文件的差异

■ dev

这是 ifconfig 的信息来源，也就是网络接口的信息，像经常使用的以太网接口 eth 也在其中。以 ifconfig 命令为例（如图 3-41 所示），其对应的文件就是 dev 这一个文件，各位读者不妨比对图 3-41 中 ifconfig 命令的结果，所框起来的数字是不是都已经存在于“/proc/net/dev”文件之中了。

```
[root@LinuxTree net]# head -4 dev;ifconfig eth0
Inter-|   Receive                                                |  Transmit
 face |bytes    packets errs drop fifo frame compressed multicast|bytes    packe
ts errs drop fifo colls carrier compressed
    lo:     896      12    0    0    0     0          0         0      896
12    0    0    0    0       0          0
  eth0: 4408997   38856    0    0    0     0          0         0   935100    64
55    0    0    0    0       0          0
eth0      Link encap:Ethernet  HWaddr 00:0C:29:FE:DC:2F
          inet addr:10.32.15.28  Bcast:10.32.15.255  Mask:255.255.252.0
          inet6 addr: fe80::20c:29ff:fefe:dc2f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:38856 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6455 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4408997 (4.2 MiB)  TX bytes:935100 (913.1 KiB)
          Interrupt:18 Base address:0x1400

[root@LinuxTree net]#
```

图 3-41：/proc/dev 文件与 ifconfig 之间的关联

■ ip_tables_X

有三个文件是与 iptables 相关的记录（这里仅限于 IPv4），分别为 ip_tables_matches、ip_tables_names、ip_tables_targets 文件（如图 3-42 所示），代表 iptables 中所存有的三种不同功能，但这当然不是 iptables 会引用的所有文件。

```
[root@LinuxTree net]# cat ip_tables_matches
state
icmp
udplite
udp
tcp
[root@LinuxTree net]# cat ip_tables_names
nat
filter
[root@LinuxTree net]# cat ip_tables_targets
MASQUERADE
DNAT
SNAT
REJECT
ERROR
[root@LinuxTree net]#
```

支持的过滤模式

支持的规则表

支持的规则炼目标

图 3-42：三个和 iptables 有关的文件

■ sockstat

此文件顾名思义就是系统 socket 的状态，最基本的信息就是 TCP 与 UDP 分别有多少个 socket 在使用中（如图 3-43 所示）。

```
[root@LinuxTree net]# cat sockstat
sockets: used 166
TCP: inuse 9 orphan 0 tw 0 alloc 13 mem 1
UDP: inuse 12
UDPLITE: inuse 0
RAW: inuse 0
FRAG: inuse 0 memory 0
[root@LinuxTree net]#
```

图 3-43：sockstat 文件中的 socket 信息

■ tcp

和刚刚的 sockstat 文件有异曲同工之妙，只是此文件其实就是 netstat 的 TCP 联机中的版本（可以使用【netstat -ntlp】命令查看）。从图 3-44 中可以知道，上方灰色框部分的两个字段分别是“local_address”和“rem_address”，主要表示“本地地址”与“远程地址”，但要特别注意的是，在这地址中不仅有 IP Address，还包括了“port number”，这点很重要。

以下方的灰色框为例，其所代表的分别是以下两个地址：

- 本地地址：“1C0F200A：0017”所表示的是“10.32.15.28：23”。
- 远程地址：“CF0F200A：05A5”所表示的是“10.32.15.207：1445”。

当然，若不想看得这么累，直接用【netstat】命令就可以看到一样的结果，这也是这些网络命令最重要的部分。

```
[root@LinuxTree net]# cat tcp
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   ui
d  timeout inode
   0: 0100007F:08A0 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6039 1 dc23d540 3000 0 0 2 -1
   1: 00000000:026F 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 5573 1 dc23d080 3000 0 0 2 -1
   2: 0164A8C0:0035 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6896 1 d9ab1ac0 3000 0 0 2 -1
   3: 017AA8C0:0035 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6884 1 da060100 3000 0 0 2 -1
   4: 00000000:0017 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6190 1 daa0a0c0 3000 0 0 2 -1
   5: 0100007F:0277 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6095 1 daa0a580 3000 0 0 2 -1
   6: 0100007F:0019 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6266 1 da060a80 3000 0 0 2 -1
   7: 0100007F:089F 00000000:0000 0A 00000000:00000000 00:00000000 00000000
0        0 6055 1 daa0aa40 3000 0 0 2 -1
   8: 1C0F200A:0017 CF0F200A:05A5 01 00000002:00000000 01:00000021 00000000
0        0 10743 4 d9ab1140 335 40 9 3 2
[root@LinuxTree net]# _
```

图 3-44：/proc/net/tcp 文件中的信息

■ udp

和刚刚的 tcp 有一样作用的文件，也都是【netstat】命令所支持的格式，唯一的差别就是这里记录的是 UDP 的联机状态。

3.2.9　/proc/scsi

如果系统上有任何的 SCSI 设备，找这个目录就一定没有错了，因为这里面有所有 SCSI 设备的信息，当然从 IDE 以后的 SATA 或 USB 接口信息，也都会一并放在这个文件之中。

图 3-45 所示是/proc/scsi 目录中的一个 scsi 文件，里面是所有接在这台计算机上，隶属于 SCSI 设备的所有基本信息，这和 dmesg 中所看到的 SCSI 信息非常相似，而在其他文件或目录中，还有更多的信息。

```
[root@LinuxTree ~]# ls /proc/scsi/
device_info  mptspi  scsi  sg
[root@LinuxTree ~]# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: ATA      Model: VMware Virtual I Rev: 0000
  Type:   Direct-Access                    ANSI  SCSI revision: 05
Host: scsi1 Channel: 00 Id: 00 Lun: 00
  Vendor: NECVMWar Model: VMware IDE CDR10 Rev: 1.00
  Type:   CD-ROM                           ANSI  SCSI revision: 05
[root@LinuxTree ~]# _
```

图 3-45：SCSC 设备在/proc/scsi 目录中的部分信息

3.2.10 /proc/sys

这个目录和系统信息其实没有太大关系（很多人会因为目录为 sys 而认为如此），它下面所放的都是一些系统核心所会使用到的一些变量，根据不同性质的文件而存放在不同的子目录中，虽然是系统核心的变量，用户可以通过【/etc/sysctl.conf】文件设置或更改其默认值。但因为这里面的值都是系统核心在使用的，所以当更改时，是实时的变更，也就是说，像家里的电灯一样（其实笔者觉得/proc/sys 下的文件，读者真的可以想成和电灯开关一样），一开就亮了，系统状态也就直接变了。

该目录下经常使用到的分类方式有以下几种。

◆ dev

在/proc/sys/下的 dev 子目录，主要存放着一些硬件的基本数据，但并不是每一个硬件都有，只有少部分的被放到这个目录中。像光驱，如果想知道这台光驱是 CDR、CDRW、DVDR 或 DVDRW，通常只能按照机器的型号自行判断，但在【/proc/sys/dev/cdrom/info】文件中有足够的信息可供用户参考。

在图 3-46 中所显示的光驱信息（0 代表不支持，1 代表支持），非常地清楚简单，一看就知道有还是没有支持，不过会有这么多的不支持，是因为这台光驱是在 VMWare 软件所仿真出的信息，并不是真的光驱（现在要买到这样稀有的光驱可能还是很贵的，因为都成为古董了）。如果各位读者检查一下手上的系统，应该会发现大部分的功能都是被支持的（都是 1）。

```
CD-ROM information, Id: cdrom.c 3.20 2003/12/17

drive name:                sr0
drive speed:               1
drive # of slots:          1
Can close tray:            1
Can open tray:             1
Can lock tray:             1
Can change speed:          1
Can select disk:           0
Can read multisession:     1
Can read MCN:              1
Reports media changed:     1
Can play audio:            1
Can write CD-R:            0
Can write CD-RW:           0
Can read DVD:              0
Can write DVD-R:           0
Can write DVD-RAM:         0
Can read MRW:              1
Can write MRW:             1
Can write RAM:             1

[root@LinuxTree dev]#
```

因为是模拟的CDROM，所以通通都不能写入

图 3-46：CDROM 的基本数据

笔者要说明的是，很多目录下的文件内容都是重复的，没有说一定要参考哪一个文件才是正确的，只要记录的是相同的信息，就应该都会是一样的结果。例如，在“/proc/sys/dev/parport/parport0/irq”文件中，所显示的 IRQ 和“/proc/interrupts”中就应该是一致的（如图 3-47 所示），所以，当用户要查询并行端口的 IRQ 时，可以使用这两个文件中的任何一个。

```
[root@LinuxTree ~]# cat /proc/sys/dev/parport/parport0/irq
7
[root@LinuxTree ~]# grep parport /proc/interrupts
  7:          0   IO-APIC-edge      parport0
[root@LinuxTree ~]#
```

图 3-47：不同文件中一样的结果

◆ fs

fs 目录中当然是一些和文件系统相关的文件，例如像文件数目、inode 数目、quota 的限制等都在其中，这些文件或许一辈子都用不到，不过要用的时候，或许是可以救命的仙丹，因为看起来都是很重要的信息，在此举几个在 man page 中曾提到，且笔者觉得有机会遇到的文件以供参考。

■ file-max

该文件的值，定义了这台系统中所有用户所使用的行程（也就是所有系统中被启动的程序）可使用的文件数量，当系统的服务越开越多，或者使用量越来越大时，有可能会造成被开启的文件大过这里面的值（笔者计算机的默认值是 50493），此时，系统就会出现报错信息。但用户可以动态调整这里面的值，只要利用【echo 100000 > /proc/sys/fs/file-max】这样的命令，就可以将数量变为 100000，以避开问题。

如图 3-48 所示，当 root 将 file-max 文件的值改变为较小值的时候（例如：10），等于用户最多只可以开启 10 个文件（但请注意，这 10 个文件包含所有执行的程序和所使用的函数库都算在内），改变之后用户会因为文件的数量已超过限制而出现报错信息。不过，root 管理员并不在此限之内，这是要特别注意的。

```
[juergen@LinuxTree test]$ cat *
-bash: start_pipeline: pgrp pipe: Too many open files in system
-bash: /bin/cat: Too many open files in system
[juergen@LinuxTree test]$ cat *
123
123
123
123
[juergen@LinuxTree test]$
```

图 3-48：file-max 文件的影响

如上所述，这也是为何刚刚笔者说有可能永远用不到 fs 目录的原因，谁知道会有这样的限制（文件被使用的数目限制），更难知道何时会遇到这样的问题，所以，若没有细看这个目录中文件的意义，遇到问题还真不容易排除（但如果是为了限制用户，这个文件就变得异常好用，对吧！）。

■ file-nr

该文件有以下三个值（如图 3-49 所示）。

Ⓐ 已被分配要处理的文件数量。

Ⓑ 没有被分配的文件数量。

Ⓒ 可被分配数量的最大值（和刚刚的 file-max 是一样的值，如果更改 file-max，这里的值也会跟着变动）。

```
[root@LinuxTree fs]# cat file-nr
1088 A 0 B    50493 C
[root@LinuxTree fs]# _
```

图 3-49：file-nr 的值

三个值都是由 kernel 动态产生的，在这文件中有 A B C 三个值，要特别注意的是当 A 值越来越大时，即可得知刚刚的 file-max 及这里的 C 值，都必须调大，不过，只须要改动 file-max 即可，因为 C 值等于 file-max 的值。

■ inode-nr

虽然这文件的两个值都是由 inode-state 文件中找出来的，不过这两个值比较重要，和刚刚的 file-max 及 file-nr 也有关系（如图 3-50），因此直接用此文件做介绍。

Ⓐ 系统已分配的 inodes 数量（nr_inodes）。

Ⓑ 可使用的 inodes 数目（nr_free_inodes）。

```
[root@LinuxTree fs]# cat inode-nr
47017 A 73 B
[root@LinuxTree fs]# cat inode-state
47017   73      0       0       0       0
[root@LinuxTree fs]# _
```

图 3-50：inode-nr 与 inode-state 的比较

当用户不断在新增文件或是删除文件时，nr_inodes 的值会跟着变动，所以，从这个值可以看出目前和 file-max 的关系是很紧密的，因为文件的数目和 inodes 的使用数目绝对是呈正比的

关系。

不过，有一点笔者自己也很困惑，在 fs 下理应有一个文件为 inode-max，这个文件会记录 inode 的最大数目，这是合理的，因为有 file-max，就应该要有 inode-max 做相对应的设定，在新版本的 Fedora 7、Fedora 8 或 SuSE10 都没有看到这个文件，这一点笔者仍在研究。

■ over_owgid 和 over_owuid

这两个文件关系到一个系统内用户和用户组的数目，换句话说，用户与用户组的数量限制就在这个文件里面。一般系统的默认值都是 65534，也就是说，当系统内的用户或用户组新增到第 65535 个“单位”的时候，系统就会认为已经超过限制而发出警讯，所以系统管理员这时就应该要先将这里的值替换掉（如图 3-51，方式和之前一样【echo 126 > /proc/sys/fs/overflowuid】。

不过，笔者认为这是有问题的，因为虽然照 proc 的文件看来，Linux 支持到 32 位的 uid 和 gid 编号（一般只到 16 位），也就是说，overflowuid 和 overflowgid 的数目应该可以到“4294967295”，但笔者尝试的结果，最大值只到 16 位的“65535”上限，结果是因为系统厂商（Fedora、Redhat or SuSE）自行限制住，所以无法调整其大小，简单来说，目前只能往下调整（其他版本要请读者自行查看），有点令人感到可惜。但如果有版本可以支持到 32 位，该用到时就千万别浪费了！

```
[root@LinuxTree fs]# cat overflow*
65534
65534
[root@LinuxTree fs]# _
```

图 3-51：overflowgid 和 overflowuid 的内容

◆ kernel

里面都是一些和 kernel 目前运行相关的文件，其文件内容都会直接影响到目前系统的运行，所以在不了解的情况下，尽量别乱动。不过，如果知道该文件的作用，有时是真的很方便的。在这个目录中有一些可以供用户“日常”使用的，在此将这些列出来供读者参考。

■ Ctrl+Alt+Del

大家都有重新启动的经验，就算不会输入【init 6】或是【reboot】命令，也知道有一个快速的重启动方式，就是【Ctrl+Alt+Del】键。

不知道大家有没有注意到，在使用【Ctrl+Alt+Del】键时，并不会直接重启，而是执行正常的重新启动“程序”，也就是会慢慢地将所有服务等都关掉后再重新启动。但如果用户很清楚

要重新启动的目的及使用方法，其实是可以不用执行这样繁复的程序而直接重新启动的。不过，在一般情况下，还是走正常的循序关机启动的流程是比较好的，如果真的需要直接重新启动，只要将这个文件的值从默认的“0”改为“1”就可以了，比其他任何的设置都要来得快（请务必记得，当设为“1”后，【Ctrl+Alt+Del】键将会直接重启，所有现行文件将都会遗失）。

■ domainname

在网络上的服务器，大部分都要设置域名，对系统来说，这就是一个 kernel 的变量，和一般所知道的环境变量或是区域变量又是分开的，用户只要直接修改此文件就可以直接换掉现行的 Domain Name，和 domainname 命令的效果相同，domainname 命令的用法如图 3-52 所示。

```
[root@LinuxTree kernel]# domainname

[root@LinuxTree kernel]# echo "test.com.tw" > domainname
[root@LinuxTree kernel]# domainname
test.com.tw
[root@LinuxTree kernel]#
```

图 3-52：domanname 文件的用法

■ hostname

这个 hostname 文件和刚刚的 domainname 是一样的意思，只是它是用来修改主机名称的，而不是域名的，所以和 hostname 命令是一样的结果，其用法如图 3-53 所示。

```
[root@LinuxTree kernel]# pwd
/proc/sys/kernel
[root@LinuxTree kernel]# hostname
LinuxTree
[root@LinuxTree kernel]# echo "test" > hostname
[root@LinuxTree kernel]# hostname
test
[root@LinuxTree kernel]# _
```

图 3-53：hostname 文件的用法

■ osrelease

该 Linux 系统的 kernel 版本，不过这里的值是不可改动的，基本上和【uname -r】命令相同。

■ ostype

这只是一个显示目前操作系统名称的文件。

■ pidmax

这个文件有点意思，它的内容数值代表的是一个系统可使用的 pid 最大的数目。PID 是系统中每一个程序的编号，换句话说，系统中软件可执行的数目会被这个文件所限制住（默认是 32768，也就是 2 的 15 次方），已经是一般系统的最大值。

不过不用担心，这数字在 32 位时仍为上限（此时要更改到更大值是会被系统拒绝的），但在 64 位的操作系统下，其值可以设定到 2 的 22 次方，依照目前硬件与软件的更新速度，相信读者手上都已经是 64 位的计算机了，所以虽然 pidmax 文件内容数值默认还是“32768”，但可以调整到更大的值。

■ pty/max 和 pty/nr

在 kernel 目录中的 pty 子目录下，有两个和之前【/dev/ptmx 文件】与【/dev/pts 目录】中所提到的 Terminal Slave 有关的文件，就是【max】和【nr】。

max 文件代表系统可接受以 Terminal 联机（像 telnet 或 ssh 之类的联机软件）所使用的接口资源数量，默认值为“4096”。因为一个联机需要一个 ptmx 所产生出的 pts，因此，max 的数目代表系统的最大联机数目。

nr 文件中的值为目前系统中的 pts 使用数量，换言之，就是目前系统所有的 Terminal 联机数量，所以 nr 文件的值无法大过 max 的值。

如图 3-54 所示，当 nr 文件的值已经等于 max 的值时，/dev/ptmx 就会无法再产生新的 /dev/pts 供远程联机使用，会造成联机错误，如图 3-54 所示。

图 3-54：因联机到达/proc/sys/kernel/pty/max 的上限而失败

■ threads-max

这文件可重要了！这是此系统同时可使用多少个 thread（线程）的定义文件，对于什么是线程，简单地说，一个程序执行后会变为内存中的进程，这是已知的，而进程会再变为多个线程

才交给 CPU 处理，所以对 CPU 来说，看到的都是线程而非进程。

这就是为何这文件很重要了，基本上不需要改动，但如果想试看看 thread 与系统执行力的关系如何，可以将这文件的值降低，就会发现系统已无法处理事情（如图 3-55 所示，但因为基本上每一个程序会用到许多 thread，所以读者在设定时，不一定要设置为“0”）。

```
[root@LinuxTree kernel]# echo "0" > threads-max
[root@LinuxTree kernel]# ls /root/
-bash: fork: Resource temporarily unavailable
[root@LinuxTree kernel]# echo "16384" > threads-max
[root@LinuxTree kernel]# ls /root/
anaconda-ks.cfg  Download      install.log.syslog  Public
Desktop          file1         Music               Templates
Documents        install.log   Pictures            Videos
[root@LinuxTree kernel]#
```

图 3-55：thread 数目太小时系统资源会无法使用

但还是提醒读者，这文件的默认值在 Redhat 网站上看到的是“2048”，不过，笔者看到几个系统（Fedora、RHEL、SuSE），每一个都是“2048”的几倍，标准各不相同，唯一可以确定的，就是该值绝对不可以太小。

■ version

这和前面的 osrelease 及 ostype 属同类型的内容，都是 kernel 版本的一部分信息，可以通过【uname -a】命令得到一样的结果（在结果的后半段），其内容如图 3-56 所示，最前面的“#1”代表，这版本的 kernel 已经从原始的 kernel source 文件编译过几次的意思，“SMP”则是 kernel 的类型，最后面接着则是被编译的时间注记。

```
[root@LinuxTree kernel]# cat version
#1 SMP Wed May 23 22:35:01 EDT 2007
[root@LinuxTree kernel]#
```

图 3-56：version 文件的内容

◆ net

该目录中存放大量 TCP/IP 协议所需的调整参数资源，在此目录中，如果可以善加利用，将可以针对网络的传输模式做一些修改，不过，目前大部分会使用到的还是 IPv4 的协议，所以比较重要的都在其中的 IPv4 目录下。

而 Linux 很多的网络相关功能是由 kernel 支持的，所以在/proc 目录下，【/proc/sys/net】就是一个专门存放 kernel 使用中的网络相关“开关”或是“变数”的地方。因为都是由 kernel 所动态产生出来的，所以部分的信息都是要在功能开启后才会产生，比如说，要通过 module（模

块）才可使用的功能。

像 bonding 的机制（可将多个网络接口合并的功能），在 Linux 中以 module 的方式存在，也就是说，当用户要使用 bonding 功能时，首要的工作就是要加载 bonding 的 module，这样才可以将该功能启动。

如图 3-57 所示，在 bonding 的模块被加载后，会由 kernel 在/proc 中（或某些其他目录）产生其所需的文件，当然这只是其中的一种功能，还有很多功能也都是如此的方式，不过在此要强调的，是当读者看到目录中有些和笔者的环境不相符时，或许就是因为某些服务、软件、模块的差异所造成的。所以，既然知道网络相关的文件是放在 net 目录下，当遇到像网络方面的问题时，就必须要想到【/proc/sys/net/】目录，另外更重要的是，通过这些目录及文件可以知道 proc 机制所含的意义。

```
[root@LinuxTree ipv4]# pwd
/proc/sys/net/ipv4
[root@LinuxTree ipv4]# ls neigh/
default  eth0  lo  virbr0
[root@LinuxTree ipv4]# modprobe bonding
bonding: Warning: either miimon or arp_interval and arp_ip_target module paramet
ers must be specified, otherwise bonding will not detect link failures! see bond
ing.txt for details.
[root@LinuxTree ipv4]# ls neigh/
bond0  default  eth0  lo  virbr0
[root@LinuxTree ipv4]# _
```

bonding模块所产生的bond0网络接口

图 3-57：目录中因加入 bonding 模块而动态新增对应之 bond0 目录

在 net 目录中主要用以下几个目录（bridge、core、ipv4、ipv6、netfilter、token-ring、unix）分类存放数据，其中以 IPv4 目录为存放最多信息的地方，因为一般网络经常使用到的协议就是 IPv4。

■ bridge

当用户将该主机建置成 bridge 时，会存放其相关信息在此目录中，读者可以在图 3-58 中看到该目录所存放的 4 个文件，每一个文件的开头都是“bridge-nf”，其实这是一个特殊工具“bridge-netfilter”的简写，这个工具主要的目的就是要担任在 bridge 下的 Firewall 角色，从 2.6 kernel 之后就变为内建的工具，所以，如果要关掉也必须将 kernel 重新编译过（在 Networking 中的 Netfilter 功能中）。

```
[root@LinuxTree bridge]# pwd
/proc/sys/net/bridge
[root@LinuxTree bridge]# ls
bridge-nf-call-arptables  bridge-nf-call-iptables
bridge-nf-call-ip6tables  bridge-nf-filter-vlan-tagged
[root@LinuxTree bridge]# _
```

图 3-58：bridge 目录中的文件

每一个文件代表不一样的功能，总括来说，“0”代表“关闭”而“1”代表“打开”，至于每个文件的功能说明如下：

- bridge-nf-call-arptables：是否要将 ARP 的数据包桥接（bridge）到 arptables 的 FORWARD 链接（chain）。
- bridge-nf-call-iptables：是否要将 IPv4 的数据包桥接到 iptables 的链接。
- bridge-nf-call-ip6tables：是否要将 IPv6 的数据包桥接到 ip6tables 的链接。
- bridge-nf-filter-vlan-tagged：是否要将含有 vlan 标头（tag）的 ARP 或 IP 数据包桥接到 arptables 或 iptables。

■ core

用来存放 TCP/IP 的主要核心参数，像其中的 rmem_default 文件内容就是记录 TCP/IP 中 Slide Window 的默认“receive window size” 值；反之，wmem_default 则是默认的“send window size”值。其他都是 TCP/IP 类似相关的文件，这里举几个例子：

- message_burst：单位为 1/10 秒，其值代表一个网络传送来的警告信息（Warning Message）可占用的时间（换句话说，在这段时间中其他信息将都被丢弃），一般可用在防止以信息方式攻击的 DDoS 模式。
- message_cost：这是一个基准值，当所收到的警告信息的 cost 值越大于 message_cost 值，就代表越可被忽略。
- netdev_max_backlog：当网络传输量比系统所可处理的数据包量来得大时，网络接口的队列（Queue）可承受的数据包数量最大值。
- optmem_max：每一个联机（Socket）所可接受的“缓冲区大小（Buffer size）”最大值。
- rmem_default：默认“receive window size”值。
- rmem_max：“receive window size”最大值。
- wmem_defaul：默认“send window size”值。
- wmem_max：“send window size”最大值。

■ ipv4

这个目录中的文件多不胜数，有几百个文件在其中，都是属于 IPv4 中 TCP 与 IP 层相关的参数文件（大部分应该说是 TCP 及 IP 所使用的全域变量），所以和应用层面较少关系，因文件

太多，仅将部分较可能使用到的在此列出。

- ip_default_ttl：这是最常看到的 IPv4 设定之一，TTL（Time To Live）定义着当数据包发送出去后，经过多少路由即视为放弃的数据包，目的是避免有封包被传送到路由错误的设备中（会造成传送的流量变为死循环），所以才会以一个基准值来确认数据包的存活状态。
- ip_forward：这是另一个大家所熟悉的设定参数，转发机制开关文件。当用户要将接口与接口之间（例如：eth0 与 eth1）的网络隧道打开，就必须将此文件的值设定为“1”才可以，默认应为“0”。
- icmp_echo_ignore_all：ICMP 协议中的 echo 指的是发送 request 数据包的操作（简单来说，就是使用命令 ping 时所发送的数据包），这里可以让系统不会做出任何 echo 的操作，但要小心使用。
- icmp_echo_ignore_broadcasts：和上面的差异在于这里只会限制对广播地址的 echo 操作，一般用户在较早的 Linux，可以 ping 一个广播地址，这样所有在同一网段上的主机都会响应，但因为这样很容易被一些 DDoS 或恶意程序所利用，因此，后来就都将此功能关闭。
- ip_local_port_range：针对 outgoing 的连接（也就是连接用系统内部对外交互的方式）限制其 port 的数量，里面的两个值代表最小连接端口号码（port number）与最大连接端口号码。在设定时，要注意因为这里的 port 指的是用户在使用的，并非所谓的 well-known ports（0-1023 就是一般 HTTP 服务所使用的 80 port），所以其值目前系统会依内存大小不同而分配（大于 128MB 时为 32768-61000；小于 128MB 则为 1024-4999 或更少）。
- tcp_fin_timeout：在关闭一个网络连接时（这里指的是 TCP/IP 的 socket），发送中断需求（FIN-WAIT-1）的系统在发送后，会在收到对方的响应并转变状态为 FIN-WAIT-2，等待下一个回应。但如果在这中间对方系统无响应或对方是恶意程序，这一条 session（就是联机）会在无法关闭的状况下占用掉系统资源，因而系统只等待一个固定的时间，而 tcp_fin_timeout 文件中的值，就是这一个“固定等待时间”的秒数。
- tcp_keepalive_time：当 TCP keepalive 功能是启动状态下，系统“每隔多少时间”会送一个 TCP 的 keepalive 信息给对方，默认为两小时（7200 秒）。
- tcp_keepalive_intvl：当“tcp_keepalive_time”的时间已到，若对方没有回应，系统应“每隔多久”发送一个 probe（检测）数据包去试探对方系统，默认为每 75 秒发送一次。

- tcp_keepalive_probes：如果 probe 发送后对方仍没有响应，“最多可发送几次”probe 数据包，默认为 9 次。
- tcp_max_orphans：这里的 orphan 指的是没有任何一个用户在使用的 TCP socket（只被系统所控制），虽然这样的情况是正常的，但当数目一多，就有可能是被恶意的程序所利用或攻击，所以必须要防止（每一个 orphan 会占用系统 64KB 的物理内存）。
- tcp_orphans_retries：当系统所控制的 socket（orphan）决定要关闭时，要“询问几次”才决定关闭（询问的机制是要确定用户是否仍然在线）。这参数在服务器很好用，因为当用户不断地进入服务器中，用户离线有可能不会通知服务器，当这种情况大量发生时，将造成服务器需要很多反复的询问操作，这时如果把“tcp_orphans_retries”的值降低，将可大幅减少服务器的重复询问操作。
- tcp_max_syn_backlog：当 TCP 开始建立联机时，会使用“三方握手”的机制，首先系统会发送“SYN”数据包，对方须回传“ACK”数据包，再从系统送出“SYN+ACK”代表完成联机。这操作看似简单，但当联机数目众多时（当开许多的 IE 浏览网页时就会有很多次三方握手的操作）。一般情况都很正常，但如果对方的系统在收到 SYN 时就已经离线，本机系统会等不到对方回传的“ACK”数据包，必须先记下目前已发送“SYN”的联机有多少，这参数的目的就是定义“已发送 SYN 数据包的联机数目最大值”为多少，目前一般的系统（大于 16MB）默认值都是 1024。
- tcp_retries1：当 TCP 联机时，须“重新传送几次”，才视为传送时发生错误并回报给网络层（OSI 第三层）。
- tcp_retries2：当“重新传送超过几次”时，会将存活中（alive）的 TCP 联机中断。

在 ipv4 目录下还有 conf、neigh、route 三个子目录，分别存放一些相关的文件。

- conf 目录：此目录中存放所有接口的信息（如图 3-59 所示），只要是系统中有的网络接口，在这下面都会有相对应的目录。不过有一个子目录比较特别，就是【all】目录，因为它存放的都是“所有接口”的默认值，也就是说，当“all”目录下的文件有所变动时，所有接口都会跟着变动。

```
[root@LinuxTree ipv4]# ls conf/
all  default  eth0  lo  virbr0
[root@LinuxTree ipv4]# _
```

图 3-59：conf 目录中的接口名称

- neigh 目录：neigh 下的接口名称和 conf 下面的应该都一样，唯一的差别是，neigh 中接口目录下的文件，都是为了要和“周围的接口”交互所使用的参数，并非本身接口所

使用的。以图 3-60 为例，当上述的 bonding 功能启动，该参数就不会产生在 conf 目录下，而是会在 neigh 目录下，因为 bonding 主要的目的就是将“周围的接口”绑在一起，所以不是本身接口所使用的。

```
[root@LinuxTree ipv4]# ls neigh/
bond0  default  eth0  lo  virbr0
[root@LinuxTree ipv4]# _
```

图 3-60：neigh 目录中的接口名称

- route 目录：所有在此目录的文件，都是和系统的 routing table 有关的参数，包含 flush、redirect 或是 error message 的设定都在此目录下。

■ ipv6

和上面的 ipv4 目录是一样的意思，存放的都是 TCP/IP 中 IPv6 协议相关的参数值，但没有像 ipv4 下的全域变量，都是一些个别文件。不过，因为 IPv6 和 IPv4 的机制差异甚大，因此，目录中的文件也有所差异，只有少许的文件在其中，在此以几个例子做说明。

- bindv6only：这参数代表是否要开启 IPv4 地址对应的功能，“0”代表开启；而“1”代表关闭。
- ip6frag_high_thresh：网络在传输时，会将 TCP/IP 的数据包切割成一个个的片段（以符合 MTU 的大小）发送，对方接收到再将收到的片段重新组合，而这片段就是 Fragment。这参数就是代表系统对“重组 Fragment 所可用的最大存储器”大小。
- ip6frag_low_thresh：这参数代表系统对“重组 Fragment 所可用的最小存储器”大小。
- ip6frag_time：针对 IPv6 的 Fragment，可“在内存中保留几秒钟”。

在 IPv6 目录中也有 conf、neigh、route 的目录，和 IPv4 下的作用是同样的分类方式，这里就不多做介绍。

另外，在 IPv6 中多了一个“icmp”的子目录，这下面也只有硕果仅存的一个文件“ratelimit”，只是为了要定义 IPv6 机制中当传送 ICMPv6 数据包时的最大速度。

■ netfilter

netfilter 是一种针对数据包过滤或检查的一种机制，最常看到的应用工具程序就是 iptables，也就是大家常用来架设防火墙的命令。其他比较少看到的还有像 QoS 机制的 tc 命令也是其中的一种。

在目录中没有太多的文件，但都是和 Netfilter 的 conntrack 机制有关的参数（如图 3-61 所示），所谓的 conntrack，其实就是 Connection Tracking。Netfilter 的机制让用户可以通过 User Space 的命令检查 kernel 内部的“Connection Tracking State”，也就是每一条网络联机的内部状态，甚至作修改。

虽然目录中存在着这些文件，可是笔者并没有找到一份有针对这些文件的介绍，虽然从名称上可做判断，但是，没有任何根据，笔者也不敢乱下断论，唯一可确定的是，当检查或修改网络联机中的数据包值或状态时，这些参数是有可能会被使用到的。

```
[root@LinuxTree netfilter]# ls
nf_conntrack_buckets             nf_conntrack_tcp_timeout_close_wait
nf_conntrack_checksum            nf_conntrack_tcp_timeout_established
nf_conntrack_count               nf_conntrack_tcp_timeout_fin_wait
nf_conntrack_generic_timeout     nf_conntrack_tcp_timeout_last_ack
nf_conntrack_icmp_timeout        nf_conntrack_tcp_timeout_max_retrans
nf_conntrack_log_invalid         nf_conntrack_tcp_timeout_syn_recv
nf_conntrack_max                 nf_conntrack_tcp_timeout_syn_sent
nf_conntrack_tcp_be_liberal      nf_conntrack_tcp_timeout_time_wait
nf_conntrack_tcp_loose           nf_conntrack_udp_timeout
nf_conntrack_tcp_max_retrans     nf_conntrack_udp_timeout_stream
nf_conntrack_tcp_timeout_close
[root@LinuxTree netfilter]#
```

图 3-61：netfilter 下的文件

■ token-ring

存放 Token Ring 网络的目前只有一个“rif_timeout”的文件，在 Token Ring 网络中，RIF 是用来储存 Routing 的信息，但因为也没有找到正式文件解释此文件，故也就不便在此作说明。

■ unix

里面也只有一个文件，就是“max_dgram_qlen”，datagram 是 TCP/IP 在封装时 IP 层数据包的名称，此文件是用来指定 Unix Domain Socket 的接收缓冲区“可接受 datagram 的最大数量”。

◆ sunrpc

sunrpc 目录（如图 3-62 所示）主要是将所有 SUN NFS（Network File System）及 SUN RPC（Remote Procedure Call）两种协议的相关参数放在其中，比较明显的例如 nfs_debug，就是可以将 NFS 的 debug 机制打开，其中的文件大部分和程序开发人员较为相关，而不是一般在调整系统所使用。

```
[root@LinuxTree sunrpc]# pwd
/proc/sys/sunrpc
[root@LinuxTree sunrpc]# ls
max_resvport  nfsd_debug  nlm_debug  tcp_slot_table_entries
min_resvport  nfs_debug   rpc_debug  udp_slot_table_entries
[root@LinuxTree sunrpc]#
```

图 3-62：sunrpc 目录中的文件

■ max_resvport

resvport 是 xprt_bindresvport 的简写，是 NFS 中一项小功能，它可以让系统从 0～1023 的连接端口号码（port number）中占用 1～800 个 port 资源，但因为大部分的标准服务协议都在 1023 之前，所以为了避免冲突发生，须将占用的编号往后移，也就是越靠近 1023 越好，所以 NFS 在此文件记录“可占用连接端口号码的最大值”。从这一个文件及下一个 min_resvport 的值，可以知道目前 NFS 所能使用的 port 已经被调整为 665～1023，1023 是这个文件的值，665 为下一个要介绍的文件。

■ min_resvport

一样是记录 xprt_bindresvport 的信息，其中的值和上一个有关，是代表“可占用连接端口号码的最小值”。

■ nfs_debug

将 NFS Client 端的 debug 功能开启，默认为关闭。

■ nfsd_debug

将 NFS Server 的 debug 功能开启，默认为关闭。

■ nlm_debug

nlm 是 NFS Lock Manager 的简写，此文件主要功能是“开启针对 NFS lock 的 debug 机制”，默认为关闭。

■ rpc_debug

将 RPC 的 debug 功能开启，默认为关闭。

■ tcp_slot_table_entries

定义“TCP”的 RPC“同时最高可以有几条联机”，默认为 16。基本上，不要任意变动这

数字，因为虽然调高可以增加同时联机的数目，但等同于系统会在同时间增加负担，有可能反而造成性能降低。

■ udp_slot_table_entries

定义“UDP”的 RPC“同时最高可以有几条联机”，默认为 16。和刚刚的 TCP 参数一样，不要任意变动此数字，有可能反而造成性能降低。

◆ vm

此目录所包含的文件类型有内存、缓冲区或缓存的管理文件，通过这些文件可以实时地针对这些储存方式做微调的操作。但此目录下的文件很多，所以笔者也只将几个看起来比较好用的文件做介绍，大部分的文件和一般在使用的 SWAP 系统有极大的关系。

■ block_dump

此文件供用户开启 block IO（块设备的输出及输入，像硬盘）的 debug 模式，当用户开启、所有程序和输出输入有关时，便会将该信息记录在 dmesg 中，供用户参考（如图 3-63 所示）。

如果用户发现磁盘有莫名的转动而想知道其原因，可以将这一个变量值更改为“1”，就可以从 dmesg 中找到其真正影响的程序。但请注意，这信息只能在 dmesg 中找到，无法在 /var/log/message 中看到，因为这是属于 kernel 内部的信息。

```
[root@LinuxTree vm]# touch /tmp/file1
[root@LinuxTree vm]# dmesg | tail -5
bash(3052): dirtied inode 2105178 (ld-2.6.so) on dm-0
touch(3052): dirtied inode 200585 (ld.so.cache) on dm-0
touch(3052): dirtied inode 1012858 (libc.so.6) on dm-0
touch(3052): dirtied inode 2105179 (libc-2.6.so) on dm-0
touch(3052): dirtied inode 689831 (locale-archive) on dm-0
[root@LinuxTree vm]# _
```

图 3-63：将 block_dump 打开后的信息

■ drop_caches

此文件主要是将内存中的 caches、dentries、inodes 清除，可以让内存多一点干净的空间，里面的值有 3 个选项可供选择，分别为 1、2、3，其效果也可以立即显示出来（如图 3-64 所示）。

“1”是将 pagecache 清空，pagecache 是实际在使用的文件的内容，正常使用情况下，pagecache 应该会比其他两个占较大的空间。

“2”是将 dentries 及 inodes 清空，dentry 是文件名称转变为 inode 时的操作，而 inode 则是

文件在系统中的实体代号，有点像用户和 UID 的对应。

“3”是将 pagecache、dentries 及 inodes 全部清空。

在范例中，因为笔者是使用一个大文件（而且图 3-65 是以 MB 为单位），所以 cache 用得比较多，如果读者是使用 ls 或 find 命令去测试内存的变化，将会发现 cache 的变化不大，反而是 buffer 的变化较大。因为 buffer 是记录文件的 metadate（也就是一些目录的清单、权限等文件或目录之属性）；而 cache 则是文件的实际内容，也就是读者在编辑文件时的内容。所以在试验时，要注意这一点，免得误会其意义。

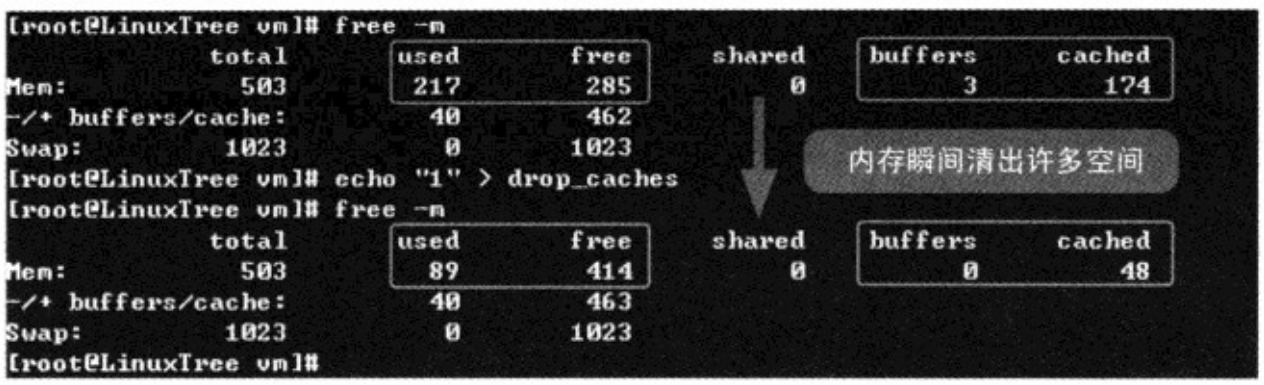


图 3-64：将 drop_caches 的值变为 1 后内存的变化

■ overcommit_memory

系统会根据此参数的设定，来决定虚拟内存的监督模式，所以会有一个内存大小的基准点。在这里的基准点数值，并非全部的内存，而是以下公式所计算出的内存大小。

SS+RAM*(r/100)＝SWAP 的大小+物理内存的大小*（overcommit_ratio/100）

其中 overcommit_ratio 的值，会在下一个文件做介绍，在 2.6 kernel 以后，overcommit_memory 参数总共支持以下三种模式。

“0”代表“尝试处理过量使用的内存”（Heuristicoverc ommit handling），这也是此参数的默认值，不过此选项检查的能力比较弱。

“1”代表“完全不检查，让它过量使用”（Always overcommit, never check），原本在 2.6 kernel 以前的版本，只要是非“0”的值，就会被归类到此设定值。

“2”代表“永远不允许过量使用”（Always check, never overcommit）。

■ overcommit_ratio

这个文件用来调整上一个文件中（overcommit_memory）所参考的公式，其中的值会直接影响系统虚拟内存的总使用量，默认为 50。

■ page_cluster

page_cluster 是针对 SWAP 在使用时，系统单次写入可同时占用的分页数目，默认值为 3。但此值的意思并不是 3 页，而是 8 页（2 的 3 次方），所以，当数字越大时，其同时间可占用的分页数量是呈指数增长。当使用环境很需要 SWAP 时，可以调整此数字，但请注意，虽然占用的数量越多会越占用 SWAP，但这和系统所存在的 SWAP 大小有关，由于文件的限制，page_cluster 超过“5”就没有意义了，因为目前集群（cluster）出的 SWAP 资料放在 32 个分页的用户组中（32-page groups），当 page_cluster 值大于 5 时，就已经超过现有的最大范围。

■ panic_on_oom

文件全名为 panic on out of memory，换句话说，若将此参数打开（设定为“1”），系统将会在内存被完全使用的情况下，进入 kernel panics 的状态。一般正常的情况，内存被占满时，系统会将一些无用的空间整理出来，但若用户不希望如此，就可以利用这个文件。

3.2.11 /proc/sysvipc

此目录下的文件，都是 System V（就是 SUN Solaris 和 SCO UnixWare 的起源）的 IPC（Interprocess Communication）所需的对象。IPC 的意思，就是系统内部各 process 间的信息交流。

IPC 的主要功能是在软件通过“monolithic process”（集合程序）的程序处理技术，变为单一线程（Single thread）的执行方式，也就是当某程序在执行时，会以一个线程的方式交给 CPU 处理，也就是以一个应用程序为基本单位。所以交互会以“应用程序对应用程序”的方式进行。

大家最常使用到的就是管道（pipe line“|”），系统默认总共会有以下 3 个文件，其内容都以简易的文件字段区分：

- msg：message queues。
- sem：semaphores。
- shm：shared memory。

3.2.12 /proc/tty

tty 这名称出自 Teletype 公司并沿用至今，史上第一个终端设备就是由这间公司生产的，也就是因为当初这间公司将其设备文件放在/dev 下面，因此，现在所有的终端设备文件，都在/dev/下并都以 tty 为命名规则，像 tty0、tty1、ttyS0、ttyS1，等等。/proc/tty 目录，则是存放有关目前可用于正在使用的 tty 设备文件信息的。

此目录中的信息其实不多，主要是 tty 设备的信息，所以都是一些像 serial port（COM port）、terminal device 或 Line discipline（PPP、SLIP、Bluetooth 等则是在 serial 接口之上，通过转换利用 serial port 做交互）的相关信息文件（如图 3-65 所示），只有两个目录加上 4 个文件，在此逐一介绍如下。

- drivers：此文件是在描述该系统中所有 tty 相关的硬件资源所使用的“driver”是哪一个，像 serial port 用的是/dev/ttyS 的资源，另外也记录每一种资源的数目有多少。
- ldiscs：ldiscs 就是 line discipline 的简写，所以此文件所记录的，都是以模拟 tty 接口为主的硬件设备，如果主机上接有相关的设备，就可以在这文件中找到相关的信息。
- driver/rfcomm：专门记录蓝牙设备的 driver 使用信息。
- driver/serial：专门记录 serial 设备的 driver 使用信息。
- ldisc/：在笔者计算机中是空目录，因为没有接任何相关的设备，如果读者的计算机上有接入设备，应该可以看到与其相关的信息。

```
[root@LinuxTree tty]# ls -l *
-r--r--r-- 1 root root 0 2008-02-21 12:39 drivers
-r--r--r-- 1 root root 0 2008-02-21 12:39 ldiscs

driver:
total 0
-r--r--r-- 1 root root 0 2008-02-21 12:39 rfcomm
-r--r--r-- 1 root root 0 2008-02-21 12:39 serial

ldisc:
total 0
[root@LinuxTree tty]#
```

图 3-65：/proc/tty 目录下的所有目录及文件

3.3　系统分类信息【/sys】

/sys 目录由一种叫 sysfs 的文件系统所建立，原本 sysfs 的名称为 ddfs（device driver file system），主要是为了将 kernel 的对象（kernel object）供一般用户使用而做的一个桥梁。这里所谓的 kernel 对象，包含 kernel 对象、对象属性及实体的关系，但这些都是 kernel 内部的信息，所以通过 sysfs 就可以呈现在用户面前（如图 3-66 所示）。

sysfs 将 kernel 的信息转换为用户可查看的文件系统结构，而将 kernel 转换成用户目录下的“基本目录（Directory）”，也就是这一节接下来要提到的每一个目录名称下所存在的子目录；“普通文件（Regular file）”就存在于每一个子目录下的项目或子项目中；至于“链接文件（Symbolic link）”，在部分子项目中（像 bus、class 等，不是每一个子项目都有）会存在，读者可在接下来的“bus”子项目中看到其范例。

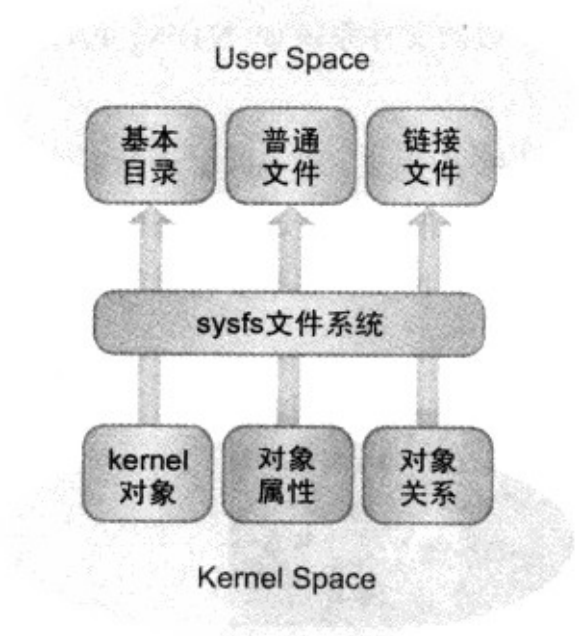


图 3-66：sysfs 和/sys 目录的关系图

之前的做法就如 proc 文件系统。而 sysfs 是因应旧有的 proc 文件系统所产生，2.6 kernel 以前的版本是看不到的，其默认主目录建立在【/sys】。sys 目录和 proc 一样都属于虚拟的文件系统，被建立在内存中，其文件系统格式为 sysfs，是在 2.6 版的 kernel 之后才被加入到正式的文件系统中。若提到是谁要加入这个文件系统，讲出来一定没人怀疑这个文件系统的重要性，因为要求加入 sysfs 机制的正是 Linux 之父——Linus Torvalds。Sysfs 被要求在 2.6 版 kernel 加入有以下几个原因：

- 为驱动设备呈现出完整的关联性。
- 制定出 hotplug 设备的办法。
- 在原本的 procfs 中有太多非 process 相关的信息。

从图 3-67 中可以看出/sys 目录和/proc 不一样，完全是以分类的方式将系统的信息存放在/sys 目录下，以方便 Linux 用户通过不同的分类找出系统相关的信息。另外，还有一点很不同的是，sys 目录规定所有文件的内容都必须是一个“单一的值”，不像以前的/proc 文件中可以有一堆的文字或是编码过的字符串在里面。

```
[root@LinuxTree sys]# ls
block  bus  class  devices  firmware  fs  kernel  module  power
[root@LinuxTree sys]#
```

图 3-67：/sys 目录下的分类方式

虽然说 sysfs 目录位于现在讨论的目录“/sys”中，但其实它在哪一个目录中并不重要。这句话听起来可能有点迷惑，但其实笔者的意思正是：“在哪一个目录都可以呈现 sysfs 的分类方

式”。因为 sysfs 是一种文件系统，因此和普通用户在挂载一般的文件系统如 NTFS、FAT、EXT2、ReiserFS 等一样，随时都可以临时使用。

如图 3-68 所示，当用户将一个暂时目录（这里以/mnt 为例）空出来，让 sysfs 文件系统使用时，只要一挂载成功，里面的文件或目录就自动被系统所导入，但这只是让用户可以通过此目录以分类方式使用 proc，并没有真正的目录或文件在其中，自然也没有要不要存盘的问题存在。

图 3-68：sysfs 被临时挂载的方式

有些读者可能会觉得，那是不是其他像 proc 比较特别的文件系统也是如此？

当然，只要是文件系统就可以挂载到目录下（但要用 mount 命令选对正确的文件系统名称唷！），这些比较特别的文件系统因为是属于虚拟文件系统，所以无需实际的硬件做对应，在使用上只要有可使用的目录可供挂载，就可像“临时租用场地”般使用该文件系统。

3.3.1　/sys/block

此目录以块设备分类（如图 3-69 所示），什么是块设备？其实就是以 block 方式（存放大小固定）读写数据的设备，例如最常使用的硬盘，其他经常看到的还有如图 3-70 列出的内存或磁盘驱动器，这些都属于块设备。

图 3-69：block 目录中的所有类型

这些目录里都有很多其子项目或针对单一项目的定义内容，例如：sda，是 SCSI 硬盘的文件代号（现在的 SATA 或 USB 也都是以此为代号），所以在 sda 的目录下，就有可能会有 sda1、sda2 或是其他 block 设备的子目录（也就是子项目）。

但不管里面有几个子目录，只要是在 block 目录中的项目，其属性的值都是固定的，也就是说，其文件数是固定的（因为一个文件只可以有一个值），都会有以下的文件，但到目前为

止，笔者并没有找到任何的官方文件注明其意义，所以仅列出笔者的看法供用户参考（文件所附之说明为笔者自己的理解）：

- dev：该设备文件的 Major 和 Minor Number，所以每一个子项目的值都会不一样。
- range：看不出其规则。
- removable：是否为可移除装置，一般的硬盘这里都会注记为“0”，也就是不可移除的；但若像 U 盘此类的装置，这里的值便为“1”，代表可移除。
- size：虽说看起来是该设备的大小，不过和实际的大小有所差异，也看不出其差异的通用性。
- stat：看不出各字段所代表的意义。
- subsystem：这是一个很特别的目录，因为在很多/sys 下的目录中，都会看到这一个目录名称，不过，它代表的并不是另一个存放其他信息的地方，里面并没有放任何数据，单纯只是让用户可以通过此目录回到“硬件分类”（像/sys/block 下或/sys/class/net 下）的子目录中，以方便搜寻。
- uevent：看不出其规则。

3.3.2 /sys/bus

这也是一个有趣的目录，用主板上的总线类别作为所有硬件的分类方式，通过总线的种类找出所要的硬件信息。举例来说，若要找系统上光驱的信息，因为它是 block 的读取方式，可以在上述的/sys/block 块设备分类中找到；但同样也可以通过光驱的总线种类找到一样的信息。在图 3-70 中（以 VMware 的 USB 光驱为例）以 block 及 bus 的分类方式，在两个不同的目录下，一样都可以找到光驱的信息。

```
[root@LinuxTree sys]# ls bus/
acpi       i2c  pci          pcmcia    pnp   serio
bluetooth  isa  pci_express  platform  scsi  usb
[root@LinuxTree sys]# cat bus/scsi/drivers/sr/1\:0\:0\:0/model
VMware IDE CDR10
[root@LinuxTree sys]# cat block/sr0/device/model
VMware IDE CDR10
[root@LinuxTree sys]#
```

图 3-70：bus 目录下的分类方式

在每一个 bus 下的子目录，都会有两个子目录：devices 和 drivers（如图 3-71 所示）。devices 是基于该分类模式下（图 3-71 中是以 pci 为分类方式），列出在其上可找到硬件的设备清单；而 drivers 则是存放在该分类模式下曾使用到的模块信息。

```
[root@LinuxTree pci]# ls devices/
0000:00:00.0  0000:00:07.0  0000:00:07.2  0000:00:0f.0  0000:00:11.0
0000:00:01.0  0000:00:07.1  0000:00:07.3  0000:00:10.0
[root@LinuxTree pci]# ls drivers/
agpgart-ali         agpgart-nvidia        ehci_hcd          piix4_smbus
agpgart-amd64       agpgart-serverworks   nptspi            serial
agpgart-andk7       agpgart-sis           ohci_hcd          uhci_hcd
agpgart-ati         agpgart-via           parport_pc        yenta_cardbus
agpgart-efficeon    ata_generic           pcieport-driver
agpgart-intel       ata_piix              pcnet32
[root@LinuxTree pci]#
```

图 3-71：bus 下子目录的两种基本分类方式

但其实在整个/sys/bus 目录中，还用了一个方式将所有的设备做进一步的关联，就是“Symbolic Link”。/sys/bus 目录将所有的设备以其总线模式分类，建立适当的 Symbolic Link 以转接到相关联的/sys 下的目录中（像/sys/class 或/sys/devices 等）。举例而言，当用户要查询 SCSI 硬盘设备时（如图 3-72 所示），可以直接通过/sys/bus 下的 Symbolic Link 查到与其相关的设备分类属性。

如图 3-72 中的第一个 link，原本是在/sys/bus/scsi/devices/0:0:0:0/block:sda 目录，其实是用链接文件的方式，帮用户切换到/sys/block/sda 目录中，也代表/sys/block/sda 的分类方式是 sysfs 另外一种的基本分类。

```
[root@LinuxTree 0:0:0:0]# pwd
/sys/bus/scsi/devices/0:0:0:0
[root@LinuxTree 0:0:0:0]# ls -l  `find . -type l`|head -5
lrwxrwxrwx 1 root root 0 2008-02-21 19:17 ./block:sda -> ../../../../../../block
/sda
lrwxrwxrwx 1 root root 0 2008-02-21 19:17 ./bus -> ../../../../../../bus/scsi
lrwxrwxrwx 1 root root 0 2008-02-21 19:17 ./driver -> ../../../../../../bus/scsi
/drivers/sd
lrwxrwxrwx 1 root root 0 2008-02-21 19:17 ./generic -> ../../../../../../class/s
csi_generic/sg0
lrwxrwxrwx 1 root root 0 2008-02-21 19:17 ./scsi_device:0:0:0:0 -> ../../../../../
../../class/scsi_device/0:0:0:0
[root@LinuxTree 0:0:0:0]#
```

主机中的硬盘

图 3-72：/sys/bus 目录下用 link 文件看原本的分类方式

3.3.3　/sys/class

class 的分类方式是用 “功能名称”（Functional type）来分类的，如 firmware、net、printer 等（如图 3-73 所示）。在这样的分类方式下，用户可以根据需求快速地找到想要了解的硬件，起码笔者感觉如此，因为分类名称完全和一般所看到的硬件名称是一样的，在硬件架构不是很熟悉的情况下，这是非常有用的。

```
[root@LinuxTree class]# ls
backlight     input             misc          scsi_disk      usb_endpoint
bluetooth     iscsi_connection  net           scsi_generic   usb_host
dma           iscsi_host        pci_bus       scsi_host      vc
firmware      iscsi_session     pcmcia_socket spi_host       vtconsole
graphics      iscsi_transport   printer       spi_transport
i2c-adapter   leds              rtc           tty
infiniband    mem               scsi_device   usb_device
[root@LinuxTree class]#
```

图 3-73：/sys/class 下的分类方式

每一种分类都有其所对应的分类对象及其属性，不过，在这样的分类（class）下，也不是里面的属性都只会存在于一个子项目中，还是要看其属性的分类。举例来说，在 scsi_disk 和 scsi_device 的子项目中，都可以看到在系统中硬盘的“型号属性”，如图 3-74 所示。

```
[root@LinuxTree class]# pwd
/sys/class
[root@LinuxTree class]# cat scsi_disk/0\:0\:0\:0/device/model
VMware Virtual I
[root@LinuxTree class]# cat scsi_device/0\:0\:0\:0/device/model
VMware Virtual I
[root@LinuxTree class]#
```

图 3-74：同一个属性可在 class 不同分类中找到

因为 class 所代表的是 kernel 中功能对象的分类，除了有些须供检查外，还可以通过里面的文件做调整，像网络接口的 MTU 大小（如图 3-75 所示）。

```
[root@LinuxTree eth0]# pwd
/sys/class/net/eth0
[root@LinuxTree eth0]# ifconfig eth0|grep MTU
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
[root@LinuxTree eth0]# echo "900" > mtu
[root@LinuxTree eth0]# ifconfig eth0|grep MTU
          UP BROADCAST RUNNING MULTICAST  MTU:900  Metric:1
[root@LinuxTree eth0]#
```

图 3-75：eth0 的 MTU 大小经文件修正后跟着改变

3.3.4 /sys/devices

devices 目录将所有在系统中被 kernel 在主板接口找到的硬件，以层级的架构一并展开，所以每一个硬件接口的目录下层，实际都是接在其接口的硬件上（如图 3-76 中 A、B、C 的阶层关系所示）。此系统的网络接口是接在 pci0000:00 下面的 0000:00:11.0 中，因此，在 devices 的阶层下，可以找到 net:eth0 的关联文件，而这个接口也可以通过 lspci 命令而取得，以证明其接口的正确性。

```
[root@LinuxTree devices]# pwd
/sys/devices
[root@LinuxTree devices]# ls pci0000\:00/ (A)                              (B)
0000:00:00.0  0000:00:07.0  0000:00:07.2  0000:00:0f.0  0000:00:11.0  uevent
0000:00:01.0  0000:00:07.1  0000:00:07.3  0000:00:10.0  power
[root@LinuxTree devices]# ls pci0000\:00/0000\:00\:11.0/
broken_parity_status  device  local_cpus  power      subsystem         vendor
bus                   driver  modalias    resource   subsystem_device
class                 enable  msi_bus     resource0  subsystem_vendor
config                irq     net:eth0 (C) rom        uevent
[root@LinuxTree devices]# lspci | grep -i pcnet32
00:11.0 Ethernet controller: Advanced Micro Devices [AMD] 79c970 [PCnet32 LANCE]
 (rev 10)
[root@LinuxTree devices]#
```

图 3-76：devices 下实体接口的阶层关系

其实在“第 3.3.3 节：sys/class”中介绍的 MTU 值，也可以在 devices 目录中找到，只要先知道该网卡的实体 pci 编号是多少（通过 lspci），再到 devices 目录中依 pci 编号，就可以找到 net:ethX 的目录，其中便有 MTU 的文件。其实这个 net:ethX 目录就是第 3.3.3 节中 class 目录下的一个对象，所以 devices 是通过关联的方式，将网络接口中的信息（如 MTU 的值）供用户查询。

不过在这样的分类下，也有例外的目录，就是 platform 和 system 两大分类目录。platform 所列举出的是一些外围设备（如图 3-77 所示），像蓝牙、电源、扬声器等；而 system 则是将非外围设备的硬件功能列出，像 CPU、ACPI 等。所以在这两个目录下，并不是按照硬件连接的顺序排列，而是依设备种类做整理的依据。

```
[root@LinuxTree devices]# ls platform/
bluetooth  floppy.0  i8042  pcspkr  power  serial8250  uevent  vesafb.0
[root@LinuxTree devices]# ls system/
acpi  clocksource  cpu  i8237  i8259  ioapic  irqrouter  lapic  timekeeping
[root@LinuxTree devices]#
```

图 3-77：platform 与 system 和其他 devices 目录的差异

3.3.5　/sys/firmware

这里的 firmware 目录，并不是真的指存放硬件 firmware（固件）的信息，而是“kernel 中的 driver”的信息。也就是说， kernel 默认支持的 driver，可在此目录中查询，像一般的网卡、硬盘或 ACPI。

不过，在 firmware 目录下的值，很多是参考 BIOS 中的 ACPI DSDT table，其实也是在参考“/proc/acpi/dsdt”文件中的数据，不过，读取这一个文件需要特殊的工具程序（要看哪一种操作系统，需要提供像 acpidump 或是 iasl 命令，若是在 SuSE 下执行 iasl，可先用“cat /proc/acpi/dsdt > /tmp/dsdt.tmp”转成一般文件，再用“iasl –d /tmp/dsdt.tmp”，就可以自动产生

一个“/tmp/dsdt.dsl”文件，此文件就可以被用户读取），才可以将该文件转换成可读的格式，若读者有机会可以看到，就会发现在 ACPI 子对象的目录下（如图 3-78 所示），还有很多子对象，而这些对象名称和刚刚从 dsdt 所看到的内容是相符的，包括 CPU、南桥或电源等信息。

```
LinuxTree:/sys/firmware # ls acpi/namespace/ACPI/
CPU0  CPU1  CPU2  CPU3  CPU4  CPU5  CPU6  CPU7  PWRF  _SB  _TZ
LinuxTree:/sys/firmware #
```

图 3-78：firmware 目录中的 ACPI 对象

3.3.6 /sys/fs

此目录应该要存放一些文件系统的 sysfs 对象信息，不过在 Redhat 及 SuSE 中都没有看到相关的文件，都只是一个空目录，不知道是不是实际上还没有加入。另外，也不知道该存放的文件，会不会就是像“/proc/sys/fs”目录中的信息，因为该目录就是存在文件系统所被定义的一些系统信息。

3.3.7 /sys/kernel

此目录所放的都是一些和 kernel 使用上直接相关的对象，不过，其中所存放的信息并不多，用到的机会也很少，因为其中大部分只和 kernel 的开发人员相关，因此，大多是关闭的（如图 3-79）。如下面两个文件：

- kexec_crash_loaded：这个文件代表系统是否开启“当 kernel crash 时将 kernel 的实时状态写入存储设备中”的功能，默认为“0”。若启动此项目，这里的值就为“1”。
- uevent_seqnum：这个文件是 udev 所产生的 event（事件）数目或序号，所以在系统加载某硬件或模块时，若符合 udev 的 rule（规则）而触发某 event，这里的“uevent_seqnum”的值就要随着增加。

```
[root@LinuxTree kernel]# ls
debug  kexec_crash_loaded  kexec_loaded  security  uevent_helper  uevent_seqnum
[root@LinuxTree kernel]#
```

图 3-79：kernel 目录中的文件清单

3.3.8 /sys/module

此目录想当然就是所有“kernel 内部”或是“被系统所加载”的模块信息存放的地方，因为这里包含所有 kernel 都会使用到的模块，所以一定会涵盖到“第 3.3.5 节：/sys/firmware”所介绍的部分。

所有这些目录下的对象名称（也就是目录名称），都和原本所加载的模块一模一样（如图 3-80 所示），有很多都是大家常看到的基本模块，所以非常好辨认，而每一个名称代表的是一个独立的目录。

```
[root@LinuxTree module]# ls -C | head -10
8250            ext3            libata          processor
ac              floppy          libiscsi        psmouse
acpi            hid             libusual        rcupdate
aerdriver       hidp            loop            rd
amd64_agp       i2c_core        lp              rdma_cm
apm             i2c_ec          mbcache         rfcomm
ata_generic     i2c_piix4       md_mod          rsrc_nonstatic
ata_piix        i8042           mii             sbs
atkbd           ib_addr         mousedev        scsi_mod
autofs4         ib_cm           mptbase         scsi_transport_iscsi
[root@LinuxTree module]#
```

图 3-80：/sys/module 下的部分文件

不过，在目录下的对象也不是每一个都已经被加载。如果是已经被加载的模组，其对象下一定都会固定有以下两个主要的部分用来表示其被加载的状态（sections 目录和 refcnt 文件），不过除了这两个部分，其他像该模块的版本、子版本或参数等也能在此（/sys/module/'module name'）目录中找到。

■ sections

这里面的文件都是代表该模块内存段（sections）的地址（如图 3-81 所示），所以每一个模块的 sections 文件可能不一样。但这可能要编写该模块的人才会对它有需要，像笔者就完全不可能用到它。有兴趣的读者不妨将 sections 中的属性和 modinfo 所带出的信息（例如 ehci_hcd 模块中的 parm）做一个比对，应该可以看出 sections 的一些端倪。

```
[root@LinuxTree module]# ls ehci_hcd/sections/
__bug_table   __param   __versions
[root@LinuxTree module]# cat ehci_hcd/sections/*
0xe08590a8
0xe0859130
0xe0859180
[root@LinuxTree module]#
```

分别代表上面三个对象属性的值

图 3-81：sections 文件中的地址信息

■ refcnt

该属性的全名为 reference count，顾名思义，就是该模块被系统其他模块所引用的次数（如图 3-82 所示）。说到这一个次数，应该很多人会想到 lsmod 命令中的引用次数，没错！这和 lsmod 中的次数是一样的。

```
[root@LinuxTree module]# cat ata_piix/refcnt
2
[root@LinuxTree module]# lsmod |grep ata_piix
ata_piix               18757  2
libata                115417  2 ata_generic,ata_piix
[root@LinuxTree module]# _
```

这里的“2”就是被多少模块所参考的意思

图 3-82：refcnt 属性中所呈现该模块被参考的次数

3.3.9 /sys/power

这个目录只存放和电源模式有关的文件，更准确的说法是只存放电源管理机制的文件。里面只有几个属性文件，属性名称（也就是文件名称）及其值都请参考图 3-83。

```
① [root@LinuxTree power]# ls
② disk ③ image_size ④ pm_trace ⑤ resume state
[root@LinuxTree power]# cat *
shutdown
524288000
0
0:0
standby disk
[root@LinuxTree power]#
```

分别代表上面 5 个对象属性的值，每一台计算机的值都有可能因环境不同而有所差异

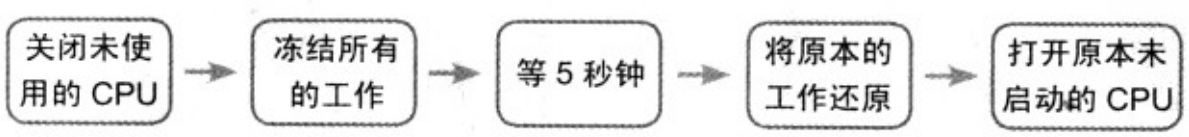
图 3-83：power 对象中的所有属性信息

每个属性的意义介绍如下：

① disk：当用户在做休眠模式 S4 的 STD（suspend to disk，或称为 Hibernate）时，要将系统的状态写入硬盘中，因此，是在定义当状态写入到硬盘后，“系统”该如何运行。目前 2.6 版的 kernel 支持以下 5 种模式：

- platform：不知其运行方式，但只有该平台支持时才可使用。
- shutdown：进入休眠时进入关机状态。
- reboot：进入休眠时进入重新启动状态。
- testproc：testproc 与下一个 test 两者的模式比较特别，都属于实时测试休眠功能所可以使用的状态，也就是让用户进入仿真的休眠模式，再自动回复到原本的系统状态，所以一直都在原来的使用状态下。

testproc 主要有以下几个步骤（系统信息如图 3-84 所示）。

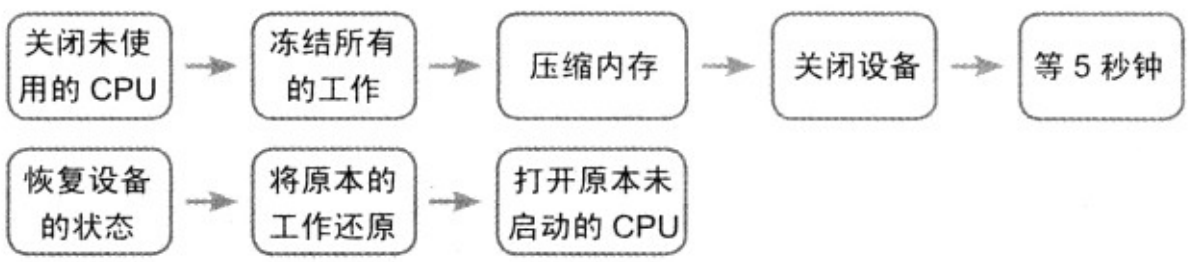


```
Stopping tasks ... done.
Shrinking memory... done (37670 pages freed)
Freed 150680 kbytes in 0.29 seconds (519.58 MB/s)
Suspending console(s)
 usbdev1.1: PM: suspend 0->1, parent usb1 already 2
 usbdev1.1_ep81: PM: suspend 0->1, parent 1-0:1.0 already 2
hub 1-0:1.0: PM: suspend 2->1, parent usb1 already 2
 usbdev1.1_ep00: PM: suspend 0->1, parent usb1 already 2
Disabling non-boot CPUs ...
swsusp: critical section:
swsusp: Need to copy 54683 pages
usb usb1: root hub lost power or was reset
Restarting tasks ... done.
Stopping tasks ... done.
swsusp debug: Waiting for 5 seconds.
Restarting tasks ... done.
Stopping tasks ... done.
swsusp debug: Waiting for 5 seconds.
```

图 3-84：使用 testproc 模式的休眠过程

- test：和前一个 testproc 的不同，是在测试时更完整，连内存和硬件设备的部分也加入模拟的阶段。

test 主要有以下几个步骤（系统信息如图 3-85 所示）。




```
Stopping tasks ... done.
Shrinking memory... done (37670 pages freed)
Freed 150680 kbytes in 0.29 seconds (519.58 MB/s)
Suspending console(s)
 usbdev1.1: PM: suspend 0->1, parent usb1 already 2
 usbdev1.1_ep81: PM: suspend 0->1, parent 1-0:1.0 already 2
hub 1-0:1.0: PM: suspend 2->1, parent usb1 already 2
 usbdev1.1_ep00: PM: suspend 0->1, parent usb1 already 2
Disabling non-boot CPUs ...
swsusp: critical section:
swsusp: Need to copy 54683 pages
usb usb1: root hub lost power or was reset
Restarting tasks ... done.
Stopping tasks ... done.
swsusp debug: Waiting for 5 seconds.
Restarting tasks ... done.
Stopping tasks ... done.
swsusp debug: Waiting for 5 seconds.
Restarting tasks ... done.
Stopping tasks ... done.
Shrinking memory... done (0 pages freed)
Freed 0 kbytes in 0.01 seconds (0.00 MB/s)
Suspending console(s)
```

这里是test及testproc两种状态下最大的差别部分，因笔者没有其他设备，因此在这里没有显示

图 3-85：使用 test 模式的休眠过程

这 5 种模式最常使用的就是“shutdown”（a 不知其作用为何，但一定要系统支持才可使用），后三者（“reboot”、“testproc”、“test”）则较少为人知，因为这 3 种大部分是在测试单位（可能也只在专门测试休眠功能的单位）才有机会被使用到，不然使用的机会少之又少，但实在是一个造福测试者的功能。不过使用上真的很简单，如笔者计算机的“disk”文件默认设定为“shutdown”，如果更改为“reboot”，当休眠完成后，将不会关机，而是“重启”。

一般谁会在休眠时下一步设置“重新启动”，这样完全无法省电，这个机制其实还是为了测试休眠模式所定义的，只是将测试的范围扩大到整个系统的流程，而不是像“testproc”和“test”仅限于测试休眠部分的功能，毕竟完整的操作是包含关机与启动。

② image_size：目前系统可接受的最大 image 大小，默认为 512 MB。

③ pm_trace：这是一个控制开关，可以决定是否要记录在多次的重新启动后，最后一次存在 RTC 中的事件点，默认值为“0”，代表不记录，若改为“1”，则可记录。

④ resume：它是当系统进行 S4（suspend to disk）时，要将系统状态写入的位置。所以这属性其实是代表某一个分区的 major 及 minor ID（一般都是 SWAP 分区），若须使用 resume 分区，就必须在启动时先在 GRUB 的开机参数中，加入 resume=/dev/xxxx 的参数，才可以让系统知道要使用该分区作为写入的分区。resume 值之所以为“0:0”，是因为主机中没有设定该参数（SuSE 在默认就会加入该参数）。

⑤ state：目前系统可支持的休眠模式，一般只会有以下 3 种（或只有其中一两种）状态（S2 一般计算机都不会支持）：

- standby：就是一般所谓的 S1，S1 是在休眠状态中最耗电的一种，CPU 中的 cache 都持续供电，但停止执行命令。在 CPU 及内存的部分都有电力在供应，但其他的装置就没有硬性规定，可断电也可供电。
- mem：就是 S3（suspend to ram），在此状态下，只有主存储器有接受供电的权利。但值得注意的是，虽然大部分的数据都会回写到内存，但硬盘本身的 Buffer 有可能来不及回写到硬盘，这样就会造成数据流失。
- disk：就是 S4（suspend to disk），俗称休眠状态（Hibernet，或称冬眠状态），其技术上的名称则为 Suspend to Disk（STD），这一阶段会将所有执行中的数据全部写入到硬盘中，而之所以要写入硬盘，就是因为要完全断电，但也因为如此，在回复到原本工作状态所使用的时间会比 S3 来得久。

更简单地说，如果将“state”文件以“disk”值写入（如：【echo “disk” > /sys/power/state】），这操作等于切换“state”开关至“disk”选项，因此，系统就会直接进入休眠，并且将休眠模式认定为前面我们提过的“disk”文件中所记载的方式，像“shutdown”。

总结

这一章只介绍 3 个目录——/dev、/proc 和/sys；但因为这 3 个目录是系统中非常重要的虚拟文件系统，没有这几个目录，用户在系统中将有如盲目般使用，对高级的用户而言是非常重要的部分。

不论虚拟文件系统的其他应用方式是什么，最基本的，就是要对系统已提供在这几个目录中的信息，了解其存放的规则，当系统要进行调整、排错、开关时，都有机会使用到这几个目录。当你对这几个目录不知道也不了解时是完全无法使用的，但如果知道它的用途，就会变得非常好用，因为有太多信息在其中。

本章并没有将“所有”的现存目录及文件一一列出，因为真的太多也太细，笔者已尽量将知道及可以查到的重要信息列出，供读者参考。

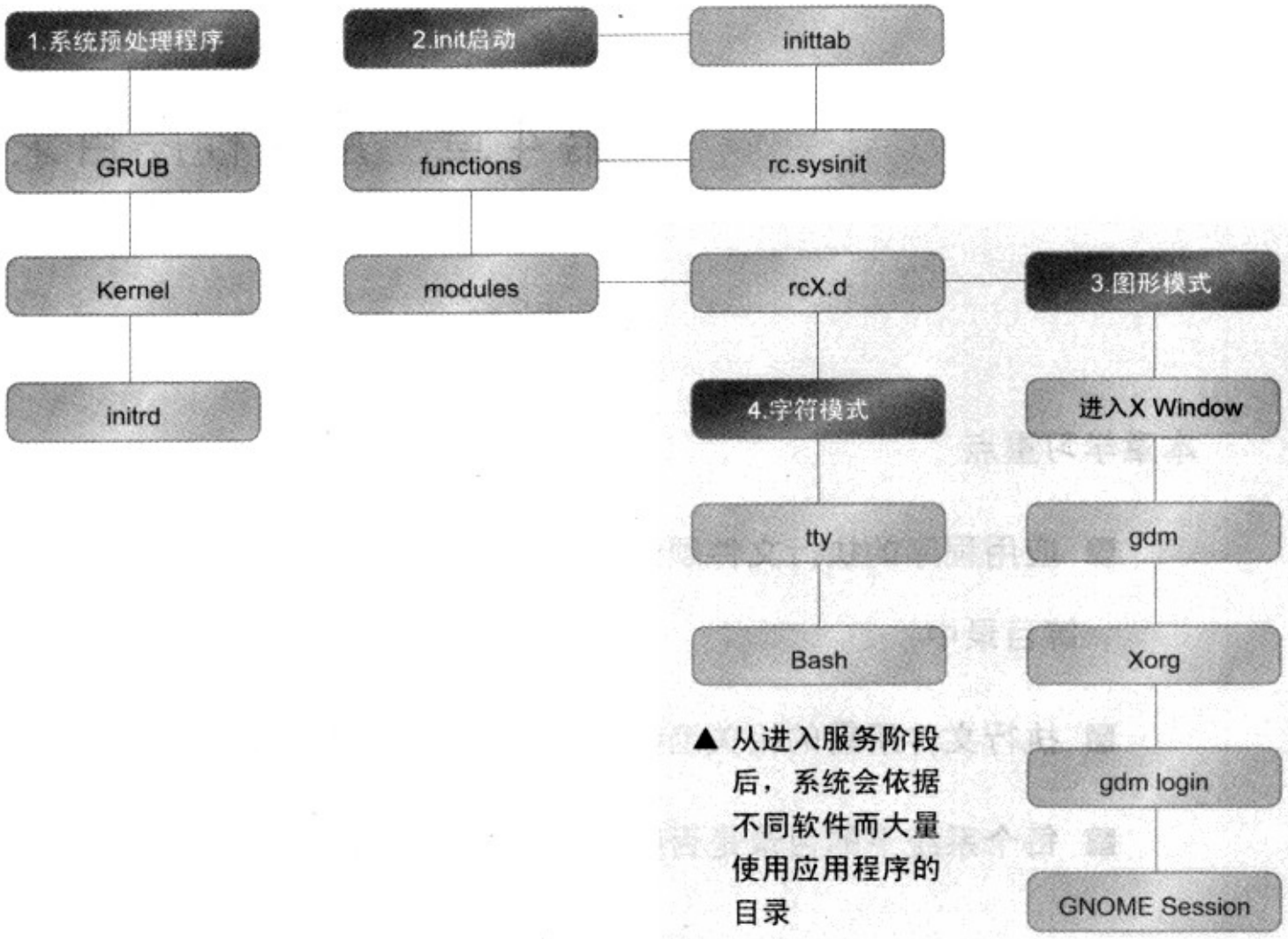
第 4 章　应用程序目录

本章学习重点

- 应用程序的执行文件须分别放在/bin、/sbin 等目录中
- 执行文件所需的相关函数或文件都放在哪些目录下
- 每个系统下的目录是否都按照标准所建立

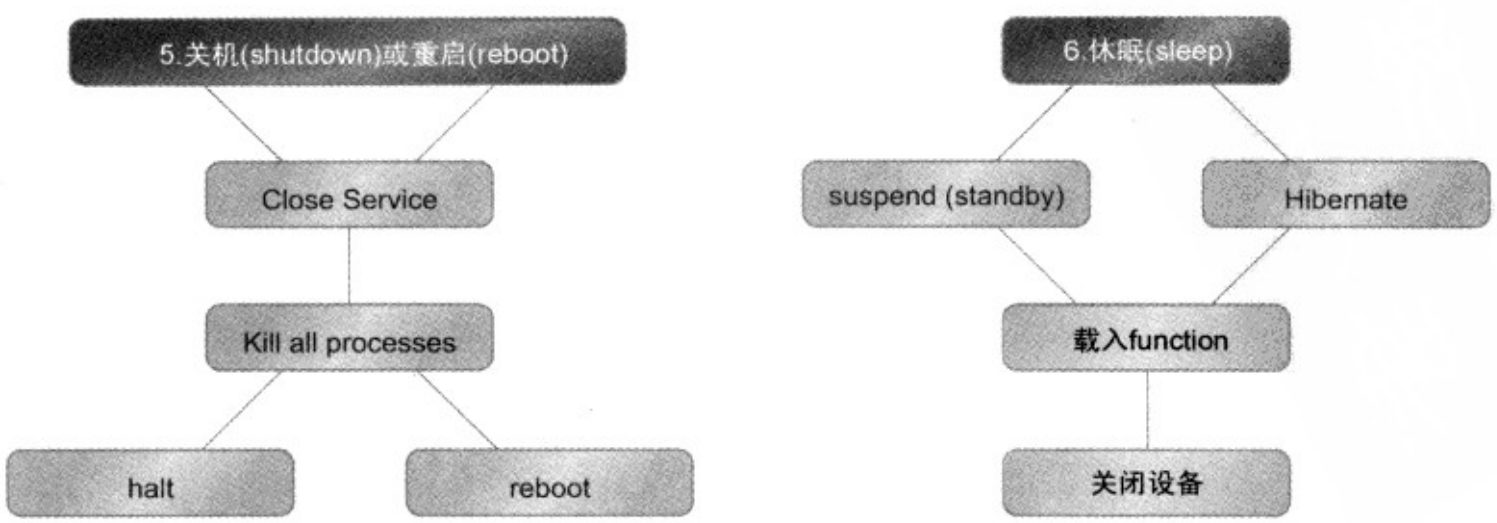
系统流程与章节内容对照图

- 启动流程



▲ 从进入服务阶段后，系统会依据不同软件而大量使用应用程序的目录

- 关机流程



本章介绍所有应用程序会使用到的目录，但请读者千万不要认为这里所说的“所有”真的是所有，因为毕竟一个程序所要执行时的范围太广，只能说应用程序所需要的静态文件，以及常需要的目录都会列在其中，至于“执行中”或不同阶段的使用范围，则要视每一个软件而定。

笔者会主要介绍一些用户日常广泛使用的目录，如/bin、/sbin、/usr、/lib 等，有些目录中的数据或许没有感觉到常在使用，但实际上，可能每分每秒都在使用，只是因为是通过第三者的程序在运作的，所以才会没有“亲身体会”的感觉，在之后的章节中只要有这样的情况，笔者都会特别提醒读者注意。

4.1 执行文件目录【/bin】与【/sbin】

/bin 与/sbin 两个目录虽然都用来存放 Linux 下的执行文件（当然有少许文件属于 link，即链接文件），但是存放的执行文件种类差异较大。另外，有两个很像的目录如【/usr/bin】与【/usr/sbin】，对这两个目录与本节所讨论的【/bin】、【/sbin】，请参考“第 4.5.1 节：/usr/bin 与/usr/sbin”的总整理表。

/bin 目录下所存放的是“所有用户”必要且可共同使用的执行文件，如“ls”、“rm”等系统默认的用户命令；而/sbin 目录下的则是只有 root 管理员或一些服务程序等级才可以使用的系统执行文件。

以图 4-1、图 4-2 为例，图 4-1 是/bin 目录下的一些文件，在其中可以看到几个常用到的命令（图中只框出了少数常用到的命令名称），这些命令都是系统中任何用户都可以使用的，所以被归类在/bin 目录下。

图 4-2 是/sbin 目录下的文件，在这个目录中会发现常用的文件少很多，这是因为大部分都是管理性质的命令，所以对一般用户或初学者而言，如果还在“纯使用”阶段，这些命令会显得非常陌生。

```
[root@LinuxTree ~]# ls /bin/
alsacard            dumpkeys  mkdir       sleep
alsaunmute          echo      mknod       sort
arch                ed        mktemp      stty
awk                 egrep     more        su
basename            env       mount       sync
bash                ex        mountpoint  tar
cat                 false     mv          taskset
```

图 4-1：/bin 下的部分文件清单

```
[root@LinuxTree ~]# ls /sbin/
accton                  ipmaddr                 nologin
addpart                 ipppd                   pam_console_apply
agetty                  ipppstats               pam_tally
alsactl                 iprofd                  pam_tally2
arp                     iptables                pam_timestamp_check
arping                  iptables-restore        parted
audispd                 iptables-save           partprobe
auditctl                iptables-xml            partx
```

图 4-2：/sbin 下的部分文件清单

但其实不论是在/bin 目录下或/sbin 目录下，大部分命令任何人都可以使用，只是/sbin 目录下的命令，一般用户的权限会比较小（如没有额外的参数可使用），所以按照使用的权限范围，分为/bin 或/sbin 的目录存放。

但 Linux 如何将/bin 及/sbin 分隔开来，不会让用户混为一谈呢？

◆ 直接调整文件权限

一般管理员要管理文件的用户权限时，最常想到的就是文件权限的调整。既然/bin 与/sbin 有些命令须要区分一般用户和管理员，这当然也是一个好办法，但如果读者仔细观察，将会发现这个办法反而是 Linux 比较少用的方式，因为 Linux 的命令原则上还是希望大家都可以用，只是默认的限制要先配置好。

如图 4-3 所示，在/sbin 下的文件，有些会将一般用户（每个文件的权限配置不一定相同，所以用户组的权限有些会开，有些则是关闭的，根据该命令的影响程度而定）的执行权限关掉（权限属性中的最后一个“x”），以避免用户误用到该命令，这也是最直接的一种方式。

```
[root@LinuxTree ~]# ls -l /sbin/auditd
-rwxr-x--- 1 root root 100180 2007-05-02 06:13 /sbin/auditd
[root@LinuxTree ~]# ls -l /bin/ls
-rwxr-xr-x 1 root root 99468 2007-04-02 23:33 /bin/ls
[root@LinuxTree ~]#
```

图 4-3：/bin 与/sbin 目录下部分文件权限的差异

◆ 限制一般用户的默认文件路径

为了让用户有机会使用到部分管理层面的命令（有些单纯的检查功能，一般用户会有使用到的机会），Linux 用了另一种方法让用户默认无法使用/sbin 目录下的命令。

当一般用户要执行属于/sbin 目录下的文件时，会发现该命令是无法被接受的（如图 4-4 所示，【ifconfig】属于/sbin 目录下的文件），追究其原因，其实并不是完全无法执行，而是因为 Linux 提供给一般用户的默认路径参数只有/bin 目录，而没有/sbin 这一路径，故该操作还未执行就已经失败了。

```
[juergen@LinuxTree ~]$ ifconfig
-bash: ifconfig: command not found
[juergen@LinuxTree ~]$ echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/usr/lib/ccache:/usr/local/bin:/bin:/usr/b
in:/home/juergen/bin
[juergen@LinuxTree ~]$
```

图 4-4：一般用户的默认路径

如果再看到 root 管理员执行的结果，一定是很正常的，因为在图 4-5 中，系统路径默认除了原本的/bin 目录外，还加入了/sbin 的路径在其中。其实读者对照图 4-4 和图 4-5，应该可以看出，只要是一般用户的执行文件相关路径，都是“bin”相关的名称；而管理员则是多加了“sbin”的相关路径。

```
[root@LinuxTree ~]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/lib/ccache:/usr/lo
cal/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
[root@LinuxTree ~]#
```

图 4-5：管理员的默认路径

其实任何用户都有机会执行到/sbin 目录下的大部分执行文件（只要该文件权限是允许其执行的），因为大部分/sbin 下的命令都有提供用户使用的权利，只是在系统的默认路径中被移除，即出现无法正常使用的情况。所以要解决的办法有两种，分别阐述如下（如图 4-6 所示）。

Ⓐ 直接用绝对路径的方式指定要使用的命令是什么。原本只输入命令的方式，用相对路径加上系统默认路径的方式去找寻该命令，但既然无法找到，就干脆给其命令完整的路径，就没有这个问题了（当然要有该命令的权限才行）。

Ⓑ 第二种方式就是将/sbin（或者其他想要加的路径）直接写入到默认路径的变量中，这样就可以在执行该命令时，让系统直接找到该命令，不过，如果希望可以一直使用下去，则最好写入如主目录的“.bash_profile”之类的用户文件中，让系统每次启动都会自动新增进去，不然重启后这一路径就必须要再加入一次。

```
[juergen@LinuxTree ~]$ /sbin/ifconfig | head -2 A
eth0      Link encap:Ethernet  HWaddr 00:0C:29:FE:DC:2F
          inet addr:10.32.15.28  Bcast:10.32.15.255  Mask:255.255.252.0
[juergen@LinuxTree ~]$ PATH=$PATH:/sbin B
[juergen@LinuxTree ~]$ echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/usr/lib/ccache:/usr/local/bin:/bin:/usr/b
in:/home/juergen/bin:/sbin
[juergen@LinuxTree ~]$ ifconfig | head -2
eth0      Link encap:Ethernet  HWaddr 00:0C:29:FE:DC:2F
          inet addr:10.32.15.28  Bcast:10.32.15.255  Mask:255.255.252.0
[juergen@LinuxTree ~]$
```

图 4-6：加入/sbin 为默认路径后的结果

4.2 函数库目录【/lib】

lib 是 library 的简写，也就是一般写程序常会使用到的“函数库”，这里的确是属于函数库的“家”。不过，在 Linux 下的/lib 目录中，并非只有“函数库”，里面还有一些其他的文件，如用户认证所须引用的文件、硬件的模块等（如图 4-7 所示），所以/lib 与其说是函数库目录，不如说是一个共享的软件仓库，只要是供所有程序引用的文件，都可以归类到其中。

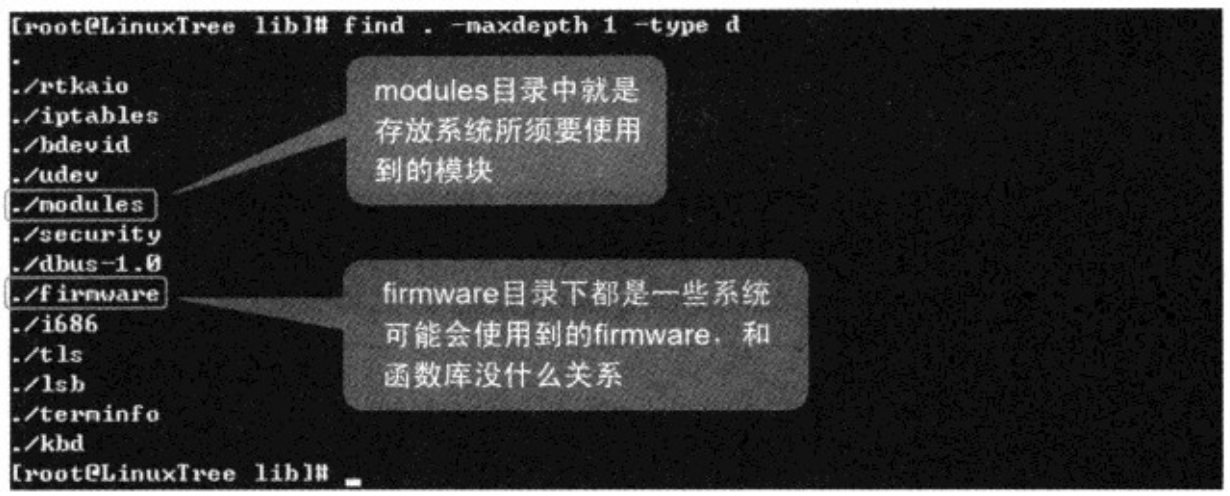


图 4-7：/lib 目录中的子目录

基本上，/lib 也可以说只要是函数库就保存在其中，正确地说，要属于系统 kernel 启动所使用的函数库，或者当执行一些在/bin 或/sbin（定义上指的是 root filesystem 下的目录，如/bin、/boot、/dev、/etc 等几个主要的目录）中的命令时会用到的函数库，才会归类到/lib 的目录下，其他开发或软件用的则会放到/usr/lib 下。

直接放在/lib 目录下的文件，都是一些很直观即单纯系统会直接使用的函数，或者某些软件在加载前所需要的函数库（加载后可能会使用子目录中的函数库），如图 4-8 中所看到的文件清单，可以看出大部分都是一些基本的系统组件，即大都是供系统本身所使用的，很少有额外安装的软件会将其函数库直接放在/lib 主目录下，避免让用户混淆。接下来，就看/lib 中的子目录到底都是用来干什么的，有些目录或许会让读者觉得很意外。

```
[root@LinuxTree lib]# rpm -qf * | head -10
nash-6.0.9-5
cpp-4.1.2-12
dbus-1.0.2-4.fc7
udev-106-4.fc7
glibc-2.6-3
iptables-1.3.7-2
kbd-1.12-21
glibc-2.6-3
glibc-2.6-3
redhat-lsb-3.1-14.fc7
[root@LinuxTree lib]#
```

图 4-8：/lib 目录下的部分文件

4.2.1 /lib/bdevid

本目录中的文件默认为由 nash 这个 RPM 文件所提供，其中放了一些 block device 的函数库，但因为 nash 目前是 Redhat（或其他操作系统）在启动时的引导者（initrd 开始加载的起头），不过，并非每一个操作系统都使用 nash，如 SuSE 所用的就是 bash，所以也并非每一个操作系统都会有这一目录存在，如图 4-9 所示。

```
[root@LinuxTree bdevid]# ls
ata.so  scsi.so  usb.so
[root@LinuxTree bdevid]#
```

图 4-9：bdevid 目录中的函数库

或许读者会觉得奇怪，nash 既然是在一启动时所使用的，也就是说，只在 initrd 阶段才会用到 nash，那为什么在实体的系统中要存放 nash 的相关函数库。但因为在系统中也可以进入“nash”的 shell（如图 4-10 所示），进而使用 nash 所提供的功能命令，所以当然会有机会使用到一些/lib 目录中的文件。

```
[root@LinuxTree ~]# nash
(running in test mode).
Red Hat nash version 6.0.9 starting
```

图 4-10：从 bash 环境进入 nash 的 shell

4.2.2 /lib/firmware

本目录下存放的不是一般的“函数库”，而是一些可能会被使用到的“firmware”（固件），有些 hotplut 硬件在安装到系统时，会被要求安装 firmware，这时该 firmware 默认会被放在这个目录下（笔者系统默认有的 firmware，如图 4-11 所示）。当然不是每一个硬件的 firmware 都会预先存在系统中，所以，当安装某些硬件时，有些需要另外安装 firmware 文件，这时就有可能会被要求将该 firmware 放入此目录下（/lib/firmware），如果放错路径，就有可能造成硬件驱动的问题。

```
[root@LinuxTree firmware]# ls
ipw2100-1.3.fw      ipw-2.4-boot.fw         ipw-2.4-sniffer_ucode.fw
ipw2100-1.3-i.fw    ipw-2.4-bss.fw          iwlwifi-3945.ucode
ipw2100-1.3-p.fw    ipw-2.4-bss_ucode.fw    LICENSE.ipw2100
ipw2200-bss.fw      ipw-2.4-ibss.fw         LICENSE.ipw2200
ipw2200-ibss.fw     ipw-2.4-ibss_ucode.fw   zd1211
ipw2200-sniffer.fw  ipw-2.4-sniffer.fw
[root@LinuxTree firmware]#
```

图 4-11：firmware 下默认存在的 firmware 文件

4.2.3 /lib/i686

本目录下可能会有一些子目录，主要用来分类用，其子目录中应该存放和/lib 相同文件名的函数库，因为在子目录下的函数库，其实都是经过一些修正或调校后的文件，以适用于不同的平台。

在笔者的系统中就有一个 nosegneg 的目录（如图 4-12 所示），其中的文件和/lib 下的文件名称一模一样，但如果细看其文件大小，就可以发现其中的差异。不过真正的差异在哪里，笔者找了半天也没找到比较清楚的说明，这就要让用到这些修正函数库的用户去发现了，如果说须要调校某些函数库，调校完的函数库文件应该都存放在/lib/i686 目录下。

```
[root@LinuxTree nosegneg]# pwd
/lib/i686/nosegneg
[root@LinuxTree nosegneg]# ls
libc-2.6.so  libm-2.6.so  libpthread-2.6.so  librt-2.6.so  libthread_db-1.0.so
libc.so.6    libm.so.6    libpthread.so.0    librt.so.1    libthread_db.so.1
[root@LinuxTree nosegneg]# ls -l libpthread-2.6.so
-rwxr-xr-x 1 root root 127612 2007-05-24 19:20 libpthread-2.6.so
[root@LinuxTree nosegneg]# ls -l /lib/libpthread-2.6.so
-rwxr-xr-x 1 root root 129608 2007-05-24 19:20 /lib/libpthread-2.6.so
[root@LinuxTree nosegneg]#
```

图 4-12：/lib/i686 及/lib 两个目录中文件的比较

4.2.4 /lib/iptables

这里存放的东西相当明显，就是所有 iptables 软件须要用到的函数库，如果用户要安装防火墙，就一定需要这一堆函数库，如果没有这些函数库，所有解析数据包的功能就都没有了。换个角度来看，这个目录其实就是 Linux 默认防火墙（如果是自行安装的就不一定）的功能库。

如图 4-13 所示，一般网管人员使用的 Linux 防火墙软件“iptables”，其功能都是需要这个目录下的函数库支持才有可能办到的，图 4-13 中圈选出的一些防火墙基本功能，如分类、记录、TCP、ICMP、数据包转发或拒绝等，都能够在这个目录中找到，从中可以了解该目录对 iptables 组件的重要性。

```
[root@LinuxTree iptables]# ls libipt_*
libipt_addrtype.so      libipt_length.so        libipt_rpc.so
libipt_ah.so            libipt_limit.so         libipt_SAME.so
libipt_CLASSIFY.so      libipt_LOG.so           libipt_sctp.so
libipt_comment.so       libipt_mac.so           libipt_SECMARK.so
libipt_connbytes.so     libipt_mark.so          libipt_SNAT.so
libipt_connlimit.so     libipt_MARK.so          libipt_standard.so
libipt_connmark.so      libipt_MASQUERADE.so    libipt_state.so
libipt_CONNMARK.so      libipt_MIRROR.so        libipt_string.so
libipt_CONNSECMARK.so   libipt_multiport.so     libipt_TARPIT.so
libipt_conntrack.so     libipt_NETMAP.so        libipt_tcpmss.so
libipt_dccp.so          libipt_NFLOG.so         libipt_TCPMSS.so
libipt_DNAT.so          libipt_NFQUEUE.so       libipt_tcp.so
libipt_dscp.so          libipt_NOTRACK.so       libipt_tos.so
libipt_DSCP.so          libipt_owner.so         libipt_TOS.so
libipt_ecn.so           libipt_physdev.so       libipt_TRACE.so
libipt_ECN.so           libipt_pkttype.so       libipt_ttl.so
libipt_esp.so           libipt_policy.so        libipt_TTL.so
libipt_hashlimit.so     libipt_realm.so         libipt_udp.so
libipt_helper.so        libipt_recent.so        libipt_ULOG.so
libipt_icmp.so          libipt_REDIRECT.so      libipt_unclean.so
libipt_iprange.so       libipt_REJECT.so
[root@LinuxTree iptables]#
```

图 4-13：/lib/iptables 中所存有部分的函数库文件

4.2.5 /lib/kbd

kbd 的全名为 Keyboard Input Driver，即键盘所使用的模块。一般在 Linux 系统环境下（无 X Window 环境），用户所使用的键盘布局，都须参照这一目录下的文件，所以这是目录中所有键盘必须使用到的配置。在用户进入系统前，系统会先配置键盘的语言（如中文、英文等），而这个文件在【/etc/sysconfig/】目录下的 keyboard 中（如图 4-14 所示），当该文件指定所使用的语言为“us”时（这绝对不是中文，因为那代表要使用的是中文键盘，目前大部分使用的都是英文键盘），系统便会直接使用/lib/kbd/keymaps/目录下的 us.map.gz 文件。

```
[root@LinuxTree qwerty]# pwd
/lib/kbd/keymaps/i386/qwerty
[root@LinuxTree qwerty]# cat /etc/sysconfig/keyboard
KEYBOARDTYPE="pc"
KEYTABLE="us"
[root@LinuxTree qwerty]# ls us.map.gz
us.map.gz
[root@LinuxTree qwerty]#
```

图 4-14：键盘默认的配置值

什么是 keymap？要知道何谓 keymap，可以先看图 4-14 中 us.map.gz 文件所在的上一层目录名称“qwerty”，各位读者可以试打这个目录名称，会发现它其实就是键盘上的英文字母（如图 4-15 所示）从最左上方往右连续打 6 个字母的结果，所以“qwerty”代表我们目前一般所用键盘的排列方式。而 keymap 则是键盘上按键的对应表，实际记录这一键盘中该有哪些功能。

图 4-15：一般键盘的字母排序

因此在/lib/kbd 的目录中，主要用于存放所有和用户在一般 console 下（非 X Window 环境）作业时键盘操作有关的文件，诸如字体和各国语言的对应表。

4.2.6 /lib/lsb

lsb 其实是一个名称为 LSB（Linux Standard Base）的组织所定义出的目录，目的是希望可以将/etc/rc.d/init.d/下的【functions】慢慢转变为/lib/lsb 目录下的【init-functions】文件，但到目前为止，官方（Fedora）的说法是建议不要采用这样的做法（如图 4-16 所示），而只先将一些机制放入目前的系统，否则会有一些非标准化的行为产生，这也是 Linux 最怕发生的事情。当有些操作系统采用新的方式，其他却停留在原地时，也会让许多 Linux 的用户产生很多不通用的问题。

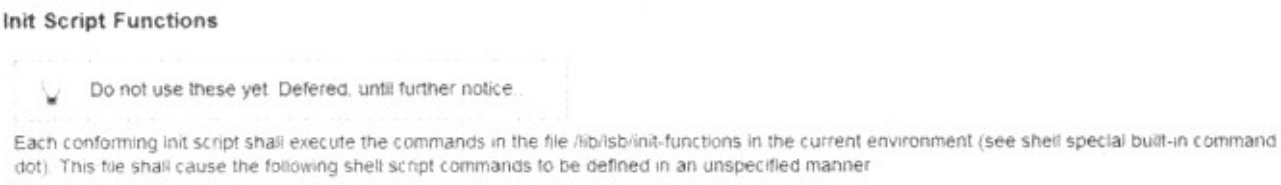


图 4-16：Fedora 官方网站的声明[1]

从新做法的文件（init-functions）内容中（如图 4-17 所示），不难看出很像之前 init 所执行的方式，和众多服务有非常密切的关系；但从目前的文件及做法来看，似乎无法直接使用，建议读者还是先用旧的启动服务方式比较保险。

至于原本的【functions】，主要就是在产生一些用户所需要的环境变量，诸如 umask、PATH 变量及参数，如【/etc/sysconfig】目录下的参数就是由这一文件所创建出来的，所以这一改革很重大且不能有错误发生，不然会直接影响到启动运作及使用的稳定性。

[1] 图片来源：*http://fedoraproject.org/wiki/FCNewInit/Initscripts*。

```
[root@LinuxTree lsb]# ls
init-functions
[root@LinuxTree lsb]# head -10 init-functions
#!/bin/sh

# LSB initscript functions, as defined in the LSB Spec 1.1.0

start_daemon () {
        /etc/redhat-lsb/lsb_start_daemon "$@"
}

killproc () {
        /etc/redhat-lsb/lsb_killproc "$@"
[root@LinuxTree lsb]#
```

图 4-17：lsb 目录中文件的部分内容

4.2.7 /lib/modules

这个目录绝对是大家每天都在使用的目录，因为里面的文件都是系统可支持的硬件模块文件（如图 4-18 所示），也就是说，该目录下缺少某一个文件就等于缺少某一个硬件一样，除非额外安装。但因为 Linux 启动时会区分系统 kernel 的版本，所以在/lib/modules 下第一层，就要先区分版本名称。然而，当用户在使用模块时，系统会自动判断目前的版本以寻找合适的模块，所以正常使用下不会有问题。

```
[root@LinuxTree modules]# ls
2.6.20-2925.9.fc7xen  2.6.21-1.3194.fc7
[root@LinuxTree modules]# modprobe -l | grep vga
/lib/modules/2.6.21-1.3194.fc7/kernel/drivers/video/vga16fb.ko
/lib/modules/2.6.21-1.3194.fc7/kernel/drivers/video/vgastate.ko
/lib/modules/2.6.21-1.3194.fc7/kernel/drivers/video/svgalib.ko
/lib/modules/2.6.21-1.3194.fc7/kernel/drivers/usb/misc/sisusbvga/sisusbvga.ko
[root@LinuxTree modules]#
/lib/modules/2.6.21-1.3194.fc7/kernel/drivers/net/forcedeth.ko
```

图 4-18：/lib/modules 下存放的 vga 模块文件

当然并不是其中的文件说加就加，假设要新增一个模块，必须通知系统有一个新的模块加入才可以使用，除非是以单一模块文件的方式加载，不然就必须事先通知系统。比如，在系统中新增了一个 addme 模块（如图 4-19 所示，其文件名为 addme.ko），但系统并不会自动知道用户将一个新的模块安插在哪一个位置，因此，系统在查询 addme 模块时（modprobe -l）无法找到；用户必须到所属的 kernel 中以【depmod -a】命令，请系统在该目录下做一次依赖性的检查，这样一来，系统就可以知道哪些模块被放到哪里，再一次查询就可以找到了。

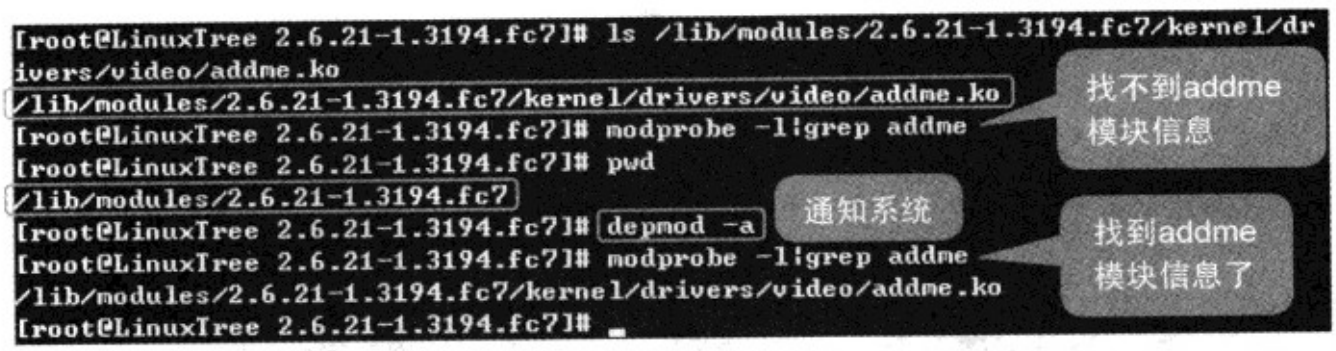


图 4-19：加入新的模块时须通知系统

另外，在模块的更新上有两个很重要的观点，在此一并提醒读者，以免在模块的制作过程中无法使用。

假设当初 kernel 已配置该模块为“*”，代表在安装 kernel 过程中已将该模块放入【ini t rd】文件中，因此，在启动时就已经先行加载，无论如何配置【/etc/modprobe.conf】，或者更改【/lib/modules】目录下的模块，都会发现所载入的模块版本还是同一个，除非读者将 initrd 文件中的模块更换，但有些模块的安装过程会帮用户考虑到这点，先行替换掉。

如图 4-20 所示，在 kernel 的 config 文件中，已经标明为预先加载的状况，因此，在【initrd】文件中一定会先放入 mptbase.ko 的模块，系统启动时在进入【initrd】阶段会先行载入。若用户是手动更新模块，将【/lib/modules】下的文件更新，并将【/etc/modprobe.conf】文件也调整过，在经过重新启动后，发现版本仍然是旧的，因为【initrd】的文件中的模块版本还是旧的。

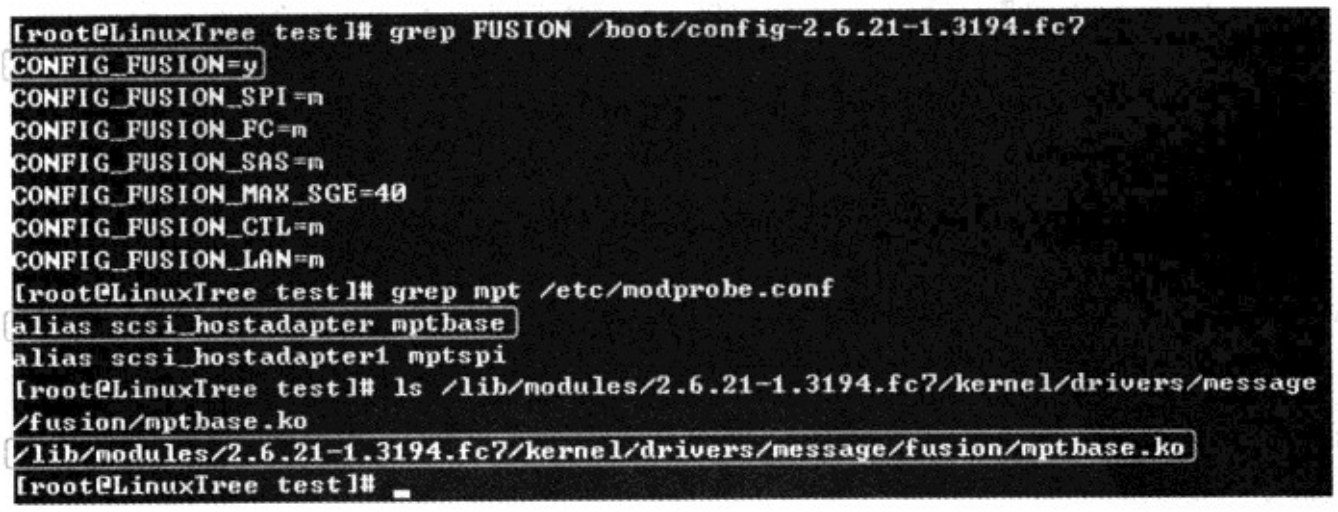


图 4-20：当 kernel 中已经默认加载该模块时的状况

另外一种情况，注意【/lib/modules/kernel-version】目录下的【updates】子目录，这个目录所代表的是经过更新后的模块所存放的区域（如图 4-21 所示）。换句话说，系统不论有多少模块或版本，在加载前会先检查【updates】子目录下有无所需要的模块，有就直接加载，因此，

即便在正确的子目录下（如“kernel/drivers/net/”）放入正确的模块文件（如 tg3），系统还是会使用【updates】子目录中的“tg3”模块。

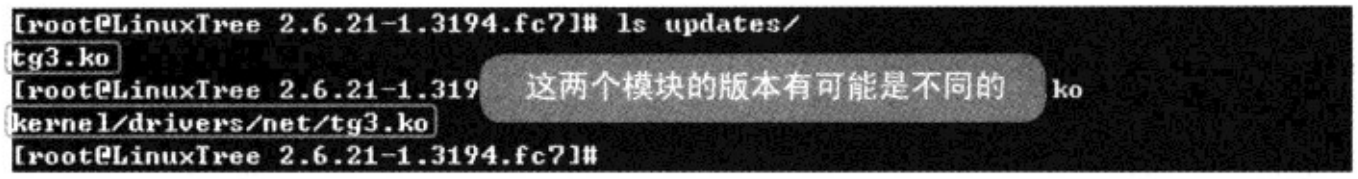


图 4-21：updates 目录和原本模块目录中有一样的模块文件

总的来说，只要抓住一个正确的模块加载顺序，就可以避免这些目录之间的问题产生（initrd 也是目录及文件的组合），如图 4-22 所示。

图 4-22：模块加载的顺序

4.2.8 /lib/rtkaio

本目录是单纯存放在 2.6 kernel 中，一种新的 I/O 读写机制 AIO（Asynchronous I/O，异步 I/O）的相关函数库，并不是只在 2.6 kernel 中才有，应该说是在 2.6 kernel 之后变为 kernel 中的标准配置。何谓异步 I/O？它的目的在于让单一的应用程序，在执行时和其他在执行中的程序相互共享一些 I/O 的作业。简单地说，以往的“同步”是让系统的 I/O 资源先让某一程序占用，在使用完毕后再释放给其他程序，这样的好处是逐一解决 I/O 问题，也不太会有错误发生。

但在以性能为主的现在，这已经不够用了。“异步”则是采取有需求就提出申请，不必等前一个结束再提，这样可以先将需求提出来，等其他的程序结束后就直接使用该资源。这和以往的同步 I/O 比起来，优点是速度绝对会快很多。

目前如 lighttpd 已经采用 AIO 的机制来存取 I/O，这一特点让许多架设网站的管理员非常兴

奋，因为经过 AIO 的增强，据说最大吞吐量（Throughput）可以提升 80%以上的性能[2]，其中一大部分是得益于 AIO 技术，对 AIO 有兴趣的读者，可以直接参考 AIO 在 sourceforge 网站的详细数据：

http://lse.sourceforge.net/io/aio.html。

4.2.9 /lib/security

这里的 security 是指 Linux 的 PAM 机制（Pluggable Authent icat ion Modules），该机制的主要目的是将所有的应用程序；以可插拔模块方式弹性地配置安全性与权限的管理。基本上，PAM 机制将所有的管理方式分为以下四类。

- account：以账号来检查其权限，比如账号是否存在、密码是否过期、某账号是否被允许使用某服务等。
- authentication：可以利用各种不同的验证方式（如 IC 卡）做身份验证，重要的就是每一种验证方式只要套用不同的模块即可，这也是 PAM 最重要的精神，“随时可插拔不同的验证模块”。
- password：这不是指一般的登录密码，而是指“更新验证机制”的密码。
- session：session 所影响到的范围，在于通信的开始与最后结束的时间点。

不论以上哪一种机制，都是调用【/lib/security】目录下的功能去完成的。然而，主要的 PAM 机制配置文件还是会放在【/etc/pam.d】目录下，而这一个【/lib/security】目录则是将所有需要的功能（函数库）集中在此，当“/etc/pam.d”配置有某功能的需求时，再直接到这目录下调用。

因为一般知道的都是配合该函数库的软件（如 login），至于这些函数库的使用，则以几个大家常会用到的 PAM 函数库为例（虽然常用到，但通常是不会知道的），因为其配置都存在【/etc/pam.d】目录中，在此只简单解释单一函数库所代表的意义与使用方式。

◆ pam_cracklib.so

不论管理员或用户都可能须要变更密码，所以须注意到一件事，就是配置密码时前后不一致或不符合密码原则将会产生错误，系统会要求用户再输入一次，若还是错误，以三次为最高次数，三次都错就放弃。

对于刚刚所提到的这些步骤，从对比密码的原则，到容许对比错误的次数，都是由

[2] 消息来源：*http://blog.lighttpd.net/articles/2006/11/12/lighty-1-5-0-and-linux-aio*。

【pam_cracklib】函数库在负责，如容许对比错误的次数就是按照【retry】的参数数值为主要根据（如图 4-23 所示），所以当【retry】的次数由“3”减为“1”的，代表只要错误发生过一次就会直接判断为失败。因此，只须对该函数库的参数或使用方式做调整，就可以很简单地修正一些系统的判断方式。

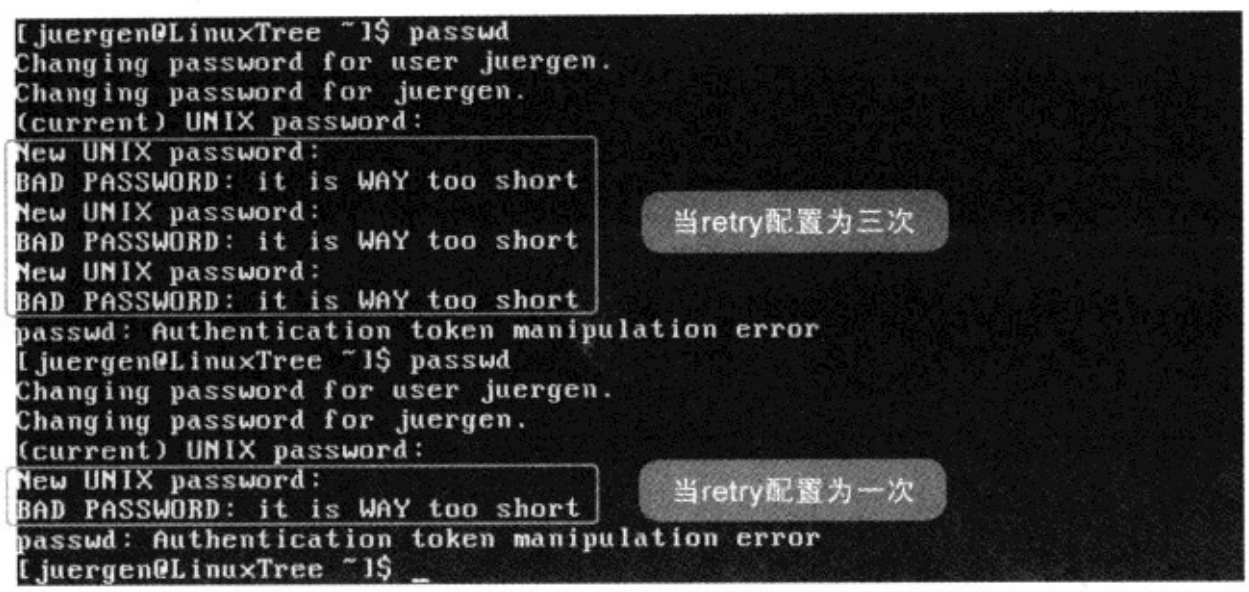


图 4-23：更改 cracklib 函数库对实际密码更动的影响

◆ pam_listfile.so

对一些服务软件的存取控制非常有帮助，此函数库的主要用意就是让管理员可以自行定义接受与拒绝服务的对象，可以支持的对象种类非常广泛，主要定义为以下 6 种：

- tty：控制台界面。
- user：本机用户。
- rhost：远程主机。
- ruser：远程用户。
- group：用户组。
- shell：用户的系统界面。

只要是系统中有机会使用到的项目，大部分都可以被 pam_listfile 所管理，如 vsftp 就是以这个函数库为默认的管理方式。

◆ pam_nologin.so

在 Linux 系统中有一种防止用户登录的配置，就是在【/etc】的目录下建立一个 nologin 文

件，这样一来，系统会自动判断为“除系统管理员 root 外，不让任何一般用户登录”，而能协助系统检测 nologin 文件及防止登录的就是 pam_nologin 函数库。

◆ pam_unix.so

大部分 Linux 用户都知道，当用户在登录时，系统账号密码的对应文件是【/etc/passwd】及【/etc/shadow】文件，但真正用这两个文件和用户所输入的账号密码去做对比的，就是【pam_unix】函数库。

◆ pam_rootok

这是一个实用而又简单的函数库，主要目的就是让 UID 为“0”的用户通过认证，换句话说，以【pam_rootok】为认证方式的软件，就只有“root”可以直接执行该软件，其他一律拒绝。比较常看到使用这个函数库做认证的软件，有【halt】、【reboot】、【eject】、【setup】、【crond】、【atd】等，不难发现，这些软件就是原本只允许 root 所使用的软件。

4.2.10 /lib/terminfo

terminfo 目录下所存放的是一般会使用到的控制台（如常见的 vt100、xterm 等）所需的函数库，当然 Linux 不只支持一种，所以在这个目录下，所有的文件是按照文件的名称开头第一个字母来存放的（如图 4-24 所示）。

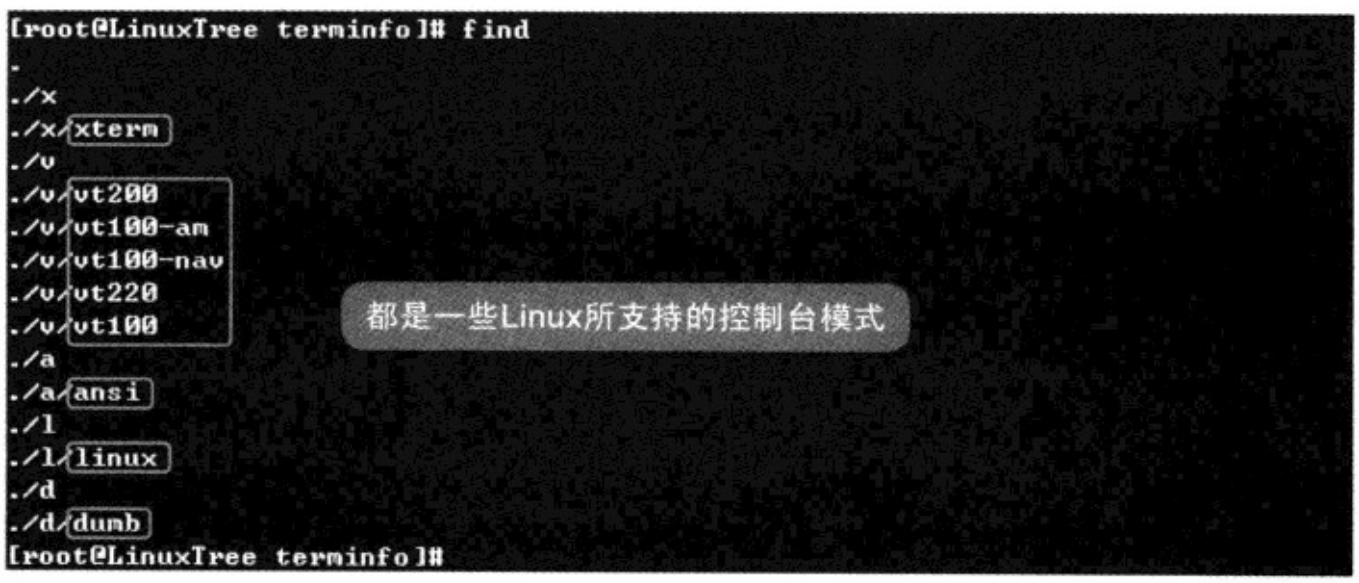


图 4-24：terminfo 目录中所存放的控制台种类

4.2.11 /lib/tls

TLS（Thread-Local Storage）是将本机内存中的程序转变为 Thread 的其中一种机制[3]，但这一功能看起来极有可能在新的操作系统中被忽略掉（笔者目前的 Fedora7 与 Fedora8 就只剩一个空目录），原因在于这种机制实现上以“segmentation”为单位，但这对目前很流行的新 Linux 架构“XEN”来说，是完全不建议使用的（如图 4-25 所示，此为 XEN 使用手册的部分内容），因为这种方式会影响到原本的使用性能。

2.5.3 TLS Libraries

Users of the XenLinux 2.6 kernel should disable Thread Local Storage (TLS) (e.g. by doing a `mv /lib/tls /lib/tls.disabled`) before attempting to boot a XenLinux kernel[4]. You can always reenable TLS by restoring the directory to its original location (i.e. `mv /lib/tls.disabled /lib/tls`).

The reason for this is that the current TLS implementation uses segmentation in a way that is not permissible under Xen. If TLS is not disabled, an emulation mode is used within Xen which reduces performance substantially. To ensure full performance you should install a 'Xen-friendly' (nosegneg) version of the library.

图 4-25：XEN 官方文件中有关 TLS 的说明[4]

但这种机制是否就没有人使用？当然不是，刚有提到，这种机制之所以在 Fedora 或 SuSE 中没有看到，是因为这些操作系统默认所采用的 Virtual Machine 的技术为 XEN 的模式，所以自然就不会采用 TLS 的函数库。

但像 Ubuntu 之类的操作系统，默认采用的是 KVM 的 Virtual Machine 解决方案，所以就会将 TLS 纳入默认的函数库中（如图 4-26 所示），要不要采用 TLS，还要看操作系统而定。

```
juergen@juergen-desktop:/lib/tls/i686/cmov$ ls
ld-2.7.so               libmemusage.so          libnss_nis.so.2
ld-linux.so.2           libm.so.6               libpcprofile.so
libanl-2.7.so           libnsl-2.7.so           libpthread-2.7.so
libanl.so.1             libnsl.so.1             libpthread.so.0
libBrokenLocale-2.7.so  libnss_compat-2.7.so    libresolv-2.7.so
libBrokenLocale.so.1    libnss_compat.so.2      libresolv.so.2
libc-2.7.so             libnss_dns-2.7.so       librt-2.7.so
libcidn-2.7.so          libnss_dns.so.2         librt.so.1
libcidn.so.1            libnss_files-2.7.so     libSegFault.so
libcrypt-2.7.so         libnss_files.so.2       libthread_db-1.0.so
libcrypt.so.1           libnss_hesiod-2.7.so    libthread_db.so.1
libc.so.6               libnss_hesiod.so.2      libutil-2.7.so
libdl-2.7.so            libnss_nis-2.7.so       libutil.so.1
libdl.so.2              libnss_nisplus-2.7.so
libm-2.7.so             libnss_nisplus.so.2
juergen@juergen-desktop:/lib/tls/i686/cmov$ _
```

图 4-26：Ubuntu 中的 TLS 相关文件

[3] 详细的说明可参考 *http://en.wikipedia.org/wiki/Thread-local_storage*。

[4] 来源出处：*http://www.cl.cam.ac.uk/research/srg/netos/xen/readmes/user.pdf*。

4.2.12 /lib/udev

在 2.6 kernel 的操作系统中，不知各位有没有发现一件事，就是当安装新的网卡时，硬件名称（ethX）不再像之前一样，永远都是固定的名称，而是会累积，有时会凭印象而用错网卡。

也就是说，若原本 A 网卡为 eth0，如果将 A 网卡换成 B 网卡，计算机中没有 eth0，只会剩一个新的 eth1。那 eth0 跑哪去了？其实是保留给原本的 A 网卡，这一机制就是所谓的 udev。

/lib/udev 目录中的文件有以下三种：

- 函数库。
- 执行文件。
- Shell Script。

其实有点像是 udev 机制的执行者，当 udev 要做任何动作时，便会调用这个目录下的文件，如刚刚提到网卡的问题，就是因为 udev 为了将网卡改名，使用/lib/udev 目录中改名的执行文件（如图 4-27 所示）。此外，还有一个大家常看到的 udev 产物，即/dev/cdrom，有兴趣的读者也可以在 udev 的 rules 目录中找找看被 udev 藏在哪了？关于这部分，本书的【/etc/udev】会有较详细的介绍。

图 4-27：/lib/udev 与 udev 机制的关系

并非所有的文件都是函数库文件，所以不是每一个 Linux 操作系统都会放在【/lib/udev】目录下，如 RHEL4 操作系统，会将使用到的 udev 相关的 script file 直接放到【/etc/udev/scripts】目录下。

4.3 还原损坏文件目录【/lost+found】

当系统在运作时，有时会无法避免宕机、断电或不正常重启动。在这样的情况下，当系统

重新启动时，发现某些文件写入未完成或其他问题产生，便会利用 fsck 程序进行文件修复动作（或手动）。而这些被修复或救回的文件，就会被放在这一个目录下，但不要期望被救回的文件和之前一样，只能说多被救回一些数据是一些，总比整个文件都没有来得好。

不过，这并不是只存在于“/”目录中，只要有一个文件系统，系统就会自动在该文件系统所在的目录下建立一个“lost+found”目录（如图 4-28 所示）。

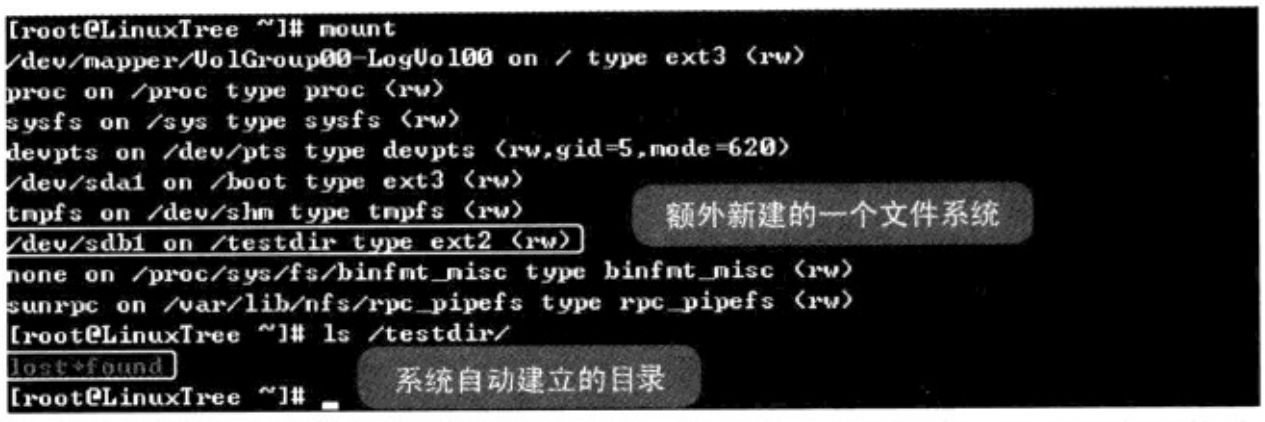


图 4-28：在文件系统下的 lost+found 目录

4.4 额外安装软件目录【/opt】

这个目录非常单纯，其实就是 option 的意思，基本上，只要是用户额外要安装的软件，就应该全部装到这一目录下。若用户并没有安装任何软件，此目录很有可能是空的。

对正式的/opt 目录用法，在其第一层的子目录只能有两种名称（如图 4-29 所示）。

- 软件名称：如 JDK、acml 等软件名称。
- 厂商名称：如 Intel、IBM 等厂商名称。

图 4-29：/opt 目录下的子目录范例

其他若有看到如/opt/doc、/opt/include、/opt/man 或/opt/lib 之类的目录，都是为了保留给管理员所使用的，别的用户若要使用，都必须到/usr 或/usr/local 下的目录，但实际上使用还是要由开发厂商决定。

4.5 用户共享目录【/usr】

在 Linux 操作系统中，一定有机会安装除操作系统本身外的一些命令或组件，但又希望让所有其他 Linux 操作系统所用，这种情况有点像是在 Linux 本身下的一个附属产品。

其具备以下几个特性：

- 可共享的目录。
- 只可读取的文件。

当这样的条件成立时，这种软件就应该安装在/usr 目录下，成为 Linux 默认操作系统目录的一个副本，让其他用户都可以在/usr 下找到原本常使用的目录，如最基本的 bin、include、lib、sbin、local、share 等（如图 4-30 所示）。但一样也可以在/usr 下建立一些额外的目录，如最常使用到就是 src 目录——用来存放 Linux kernel 文件。

```
[root@LinuxTree usr]# ls
bin  games    kerberos  libexec  sbin   src  X11R6
etc  include  lib       local    share  tmp
[root@LinuxTree usr]# _
```

图 4-30：/usr 下默认的子目录

4.5.1 /usr/bin 与/usr/sbin

在/usr 目录下的 bin 与 sbin 两个子目录，与/bin 和/sbin 的最大差别，就在于这两个目录中的执行文件都是非必要性的文件，也是一些非系统本身的软件所放置其执行文件的所在，换句话说，都是比较偏应用方面的命令。

/usr/bin 下的文件，是一般用户有机会使用到的命令，或者该软件默认就是要让所有用户使用才会放在该目录中。

/usr/sbin 下则是一些系统有可能会用到的系统命令，不过和/sbin 比起来，都是一些较次要的文件才会放在/usr/sbin 下。

光是靠文字，还是很容易将【/bin】、【/sbin】、【/usr/bin】、【/usr/sbin】四个目录的主要目的混淆，笔者在此将这四个目录整理成一份表格，希望读者可以通过表 4-1 的内容能清楚地知道每个目录的特性。

表 4-1：四个命令相关目录的比较表

	/bin	/sbin	/usr/bin	/usr/sbin
必要性命令	V	V	X	X
系统管理命令	X	V	X	V

至于【/usr/bin】、【/usr/sbin】两个目录的文件权限如何区分开，和之前提过的【/bin】、【/sbin】两个目录是一样的做法，这部分请读者自行参考“第 4.1 节：执行文件目录【/bin】与【/sbin】”中的介绍，相信会得到清楚的答案。

4.5.2 /usr/etc

虽然有些系统下有这一目录，但这是非官方标准的目录，也就是说，在正式的目录结构中，尚未开放这个目录的使用，因此，这个目录很有可能是空的（如图 4-31 所示），毕竟在 Linux 系统中，配置文件的重点一直都放在【/etc】目录上，如果要放一些配置文件的文件，目前有两种选择。

- /etc：系统主要的配置文件目录。
- /usr/local/etc：自行安装或非系统主要的配置文件目录。

```
[root@LinuxTree usr]# pwd
/usr
[root@LinuxTree usr]# ls etc/
[root@LinuxTree usr]# ls local/etc/
[root@LinuxTree usr]#
```

图 4-31：/usr/etc 与/usr/local/etc 目录默认是空的

如果用户在/usr 下有一些软件（也就是不在/usr/local 下）须要配置文件的目录，请将/usr/etc 目录干脆以 link 文件的方式直接指定到/etc 下，这样就不会有问题了。

4.5.3 /usr/games

本目录再清楚不过了，只要是电脑游戏相关的软件，就都安装到这里来吧！不过，从这个目录可以看出一件事，就是系统的文件配置方式和一般标准的目录结构，不一定完全一样。

在 Fedora7 和 Fedora8 下，该目录都是空的，也就是说，Fedora 将这些电脑游戏软件（笔者虽没玩电脑游戏，但系统总有一些默认 X Window 下的电脑游戏）大都装到/usr/bin 的目录下（感觉上也没啥不对，因为的确是“共享给大家使用的非系统命令”），不知道哪一天会不会转到/usr/games 这个标准目录下。

而且这个目录在 SuSE10 下（如图 4-32 所示），可以看到 SuSE 非常标准地将电脑游戏的执行文件都放在该目录下，所以 open source 的一个缺点就是，大家各做各的，虽然有标准，但很难贯彻执行。

```
linux:~ # ls /usr/games/
.                battlestar      dm          lbreakout2        pachi       rot13          trek
..               bcd             factor      lbreakout2server  phantasia   sail           wargames
Maelstrom        boggle          figlet      ltris             pig         showfigfonts   worm
Maelstrom-netd   briquolo        figlist     martian           pingus      snake          worms
accc             caesar          fish        mille             pingus.sh   snscore        wtf
adventure        canfield        gl-117      monop             pom         solarwolf      wump
arithmetic       cfscores        gomoku      morse             ppt         teachgammon    xlogical
armagetronad     chkfont         hangman     neverball         primes      tetris-bsd
armagetronad-stat chromium       hunt        neverputt         quiz        texmapper
atc              chromium-setup  huntd       nfs2ac            rain        torcs
backgammon       countmail       khunphan    nfsperf           random      trackballs
banner           cribbage        lbreakout   number            robots      trackgen
linux:~ #
```

图 4-32：在 SuSE 下/usr/games 目录中的文件清单

4.5.4 /usr/include

这个目录虽然在/usr 下，但所存放的文件都是一些系统中用户所会使用到的 C 语言 header 文件，如果看这个目录下的文件名，你会发现全都是扩展名为“.h”的文件（如图 4-33 所示）。

```
[root@LinuxTree usr]# ls -C include/ --color| head -5
acl                     idn-int.h               pi-header.h
af_vfs.h                ieee754.h               pi-hinote.h
aio.h                   ifaddrs.h               pi-inet.h
aliases.h               initreq.h               pi-macros.h
alloca.h                inttypes.h              pi-mail.h
[root@LinuxTree usr]#
```

图 4-33：/usr/include 下的部分文件

4.5.5 /usr/kerberos

kerberos 是一种安全机制，让用户可以直接使用支持 kerberos 机制系统上的部分资源（使用上有点类似 IE 所使用的凭证概念），只是在此，该目录有点要注意的（以 Fedora 为例）倒不是/usr/kerberos 本身，而是在其下的子目录（如图 4-34 所示）。

一般安装的软件会将其执行文件或 man page（使用手册）放在/usr/sbin、/usr/bin 或/usr/share/man 目录下，但 kerberos 把这些都放到/usr/kerberos 下的 bin、sbin 或 man 等目录中，所以和一般的目录架构概念有些许不同。

```
[root@LinuxTree usr]# pwd
/usr
[root@LinuxTree usr]# ls kerberos/
bin  man  sbin  share
[root@LinuxTree usr]# ls kerberos/bin/
kdestroy  kinit  klist  kpasswd  krb524init  krb5-config  ksu  kvno  sclient
[root@LinuxTree usr]#
```

图 4-34：/usr/kerberos 下的目录结构

在实际使用上，对用户来说没有太大的影响，因为系统在安装时已经默认将这些 kerberos 会使用到的目录加到用户默认的使用路径中，所以同样可以直接使用 kerberos 相关的文件（如图 4-35 所示）。

```
[root@LinuxTree sbin]# echo $PATH
/usr/lib/qt-3.3/bin:/usr/kerberos/sbin:/usr/kerberos/bin:/usr/lib/ccache:/usr/lo
cal/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin:/root/bin
[root@LinuxTree sbin]#
```

图 4-35：root 默认的系统路径中 kerberos 的信息

4.5.6 /usr/lib

这里大部分存放一些函数库、执行文件及连接文件，比较特别的是，存放在这里面的文件都是不希望直接被用户或 shell script 所使用的文件，而在这个目录下也可被定义为各软件新增其专属的目录空间使用，以免混淆。

如图 4-36 所示，在【/usr/lib】中有非常多的目录，因为每一个软件都有其各自所需的函数库，既然不能放在【/lib】中，各软件就会在【/usr/lib】中建立自己的子目录，再将所需的函数库全部放在里面，可以想象，在这个目录中会有非常多的文件，如笔者系统中的文件数目大约为 42 882 个（要扣除掉所有的目录），如图 4-37 所示，这个数目也是很惊人的。

```
[root@LinuxTree lib]# find . -maxdepth 1 -type d| head -10
.
./java-1.4.2
./java-1.4.0
./gcc
./jack
./alchemist
./valgrind
./tc
./java-1.5.0
./java-1.6.0
[root@LinuxTree lib]#
```

图 4-36：/usr/lib 下软件自行建立的目录

```
[root@LinuxTree lib]# find . | wc -l
42882
[root@LinuxTree lib]#
```

图 4-37：笔者系统中/usr/lib 下大致的文件数目

4.5.7 /usr/libexec

在 Redhat 的操作系统中，不知为什么在/usr 下多了一个 libexec，而这个目录下的文件夹，原本应该是属于/usr/lib 下的，因为虽然/usr/libexec 和/usr/lib 中的文件夹类型有些差异（如图 4-38 所示），不过在/usr/lib 目录中原本就可以储存这一类文件，但 RedHat 却将其搬到/usr/libexec 下，这对一些按照目录结构所写的程序在跨平台运作时可能会产生一些问题，因为原本某些执行文件路径若设为/usr/lib，在 Redhat 系统下会找不到文件。

```
[root@LinuxTree libexec]# file /usr/lib/*.so | head -5
/usr/lib/libacl.so:                          symbolic link to `../../lib/libac
l.so'
/usr/lib/libakode_alsa_sink.s                :hared object, Int
el 80386, version 1 (SYSV), s
/usr/lib/libakode_mpc_decoder                :hared object, Int
el 80386, version 1 (SYSV), s
/usr/lib/libakode_oss_sink.so                :hared object, Int
el 80386, version 1 (SYSV), stripped
/usr/lib/libakode_polyp_sink.so:             ELF 32-bit LSB shared object, Int
el 80386, version 1 (SYSV), stripped
[root@LinuxTree libexec]# file * | head -3
accessx-status-applet:                       ELF 32-bit LSB executable, Intel 80386, ver
sion 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.9, strip
ped
at-spi-registryd:                            ELF 32-bit LSB executable, Intel 80386, ver
sion 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.6.9, strip
ped
awk:                                         directory
[root@LinuxTree libexec]#
```

大部分在/usr/libexec中的函数库文件和/usr/lib中是不同类型的，但正常是可以全部放在/usr/lib下

图 4-38：/usr/libexec 与/lib 之间的差异

4.5.8 /usr/local

刚刚提到所安装的共享软件安装在/usr 下，但最好的方式是安装在/usr/local 下，因为按照 Linux 目录的标准结构而言，/usr/local 就是为了这些软件所存在的目录，除非已经存在于/usr 下的软件要进行更新之类的操作，否则，新建的软件都应该放在/usr/local 下。因此，该目录下有一些是必要的目录（当然，如果没有安装任何软件，就没有必要可言，只是默认一定会有该目录存在），在此列出几个经常被软件使用到的目录。

- bin：存放软件执行文件的目录。
- sbin：同样存放软件执行文件的目录，但此目录专门针对系统所使用的文件（和/bin 及/sbin 的意思是一样的）。
- lib：软件相关的函数库。
- share：当文件性质不好归属时就会放在此，一般会将使用手册（manual）放在这，但其实应该是要放在/usr/local/man 下。

- src：所安装软件的 source code 都放在此。

4.5.9 /usr/share

此目录都是一些共享信息，最常被用到的就是/usr/share/man 这个目录，因为都是存放各式各样的使用手册。要特别注意，/usr/share 里的信息是跨平台的，不管哪一种硬件平台的操作系统信息，都可以被允许放到这个目录下，所以信息会非常多且完整。

不过，这个目录中的内容须要随着操作系统版本的演变而变动，所以，同样的操作系统版本不同，有可能这个目录下所存放的信息也不同，像刚刚提到的/usr/share/man 目录就是一个例子，新版本的操作系统势必会多出一些功能或命令，这时就需要一些新的 man page 给用户参阅。

在/usr/share 下的目录太多太多，大部分都是一些软件各自所需共享的文件夹，在此就不赘述（因为不是一个通用规则）。不过，man page 是一个 Linux 下通用的文件格式，因此，提一下/usr/share/man 目录的大致结构。

man page 分为 8 个主要章节（如表 4-2 所示），每一章节都有各自所阐述的重点，Linux 将所有归属于该章节负责的文件放置到正确的路径下（如 ls 命令的 man page 就会丢到第 1 章的/usr/share/man/man1 目录中）。

表 4-2：man page 的章节整理表

章节	说明文件目录名称	主要内容
1	/usr/share/man/man1	一般用户所须知道命令用法都会包含在这一章中
2	/usr/share/man/man2	有关系统 kernel 的 system calls 用法，专为对程序设计有兴趣的人而写
3	/usr/share/man/man3	阐述一些函数库与 subroutines 的用法，专为对程序设计有兴趣的人而写
4	/usr/share/man/man4	一些特殊文件的说明，大部分都是像/dev 目录下的设备文件信息，或者网络界面模块的相关信息也会放在这一章
5	/usr/share/man/man5	以文件的格式为主，所以大部分都是/etc 目录下的配置文件格式，但如程序执行所产生出的格式或系统文件格式也都在其中
6	/usr/share/man/man6	凡是电脑游戏、展示，或者试用软件的说明都在此
7	/usr/share/man/man7	主要存放“杂项”，无法分类或模糊地带的文件就放在这一章
8	/usr/share/man/man8	系统管理相关功能面的说明

当然，现在 man page 也是有多国语言的表现方式，所以，如果要看中文的 man page 当然不成问题（如图 4-39 所示），只要用户将 man page 的路径指定到中文版的路径下，就可以在 X Window 下（一般 console 不支持中文，会变成乱码）正常地以中文形式查看该说明文件，对看英文比较吃力的用户有好处。

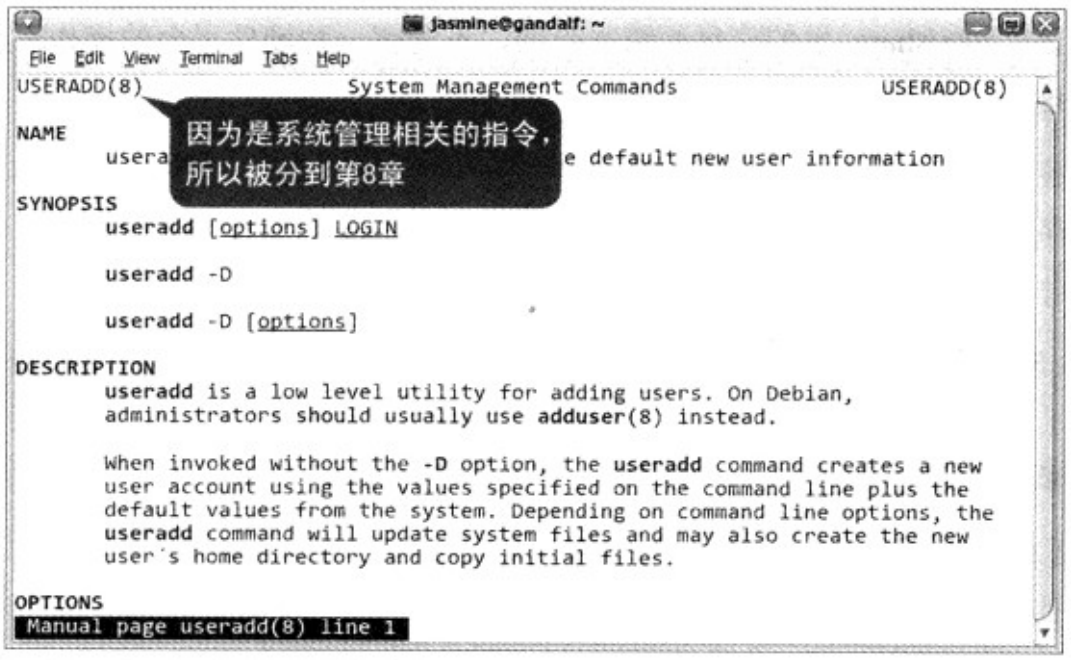


图 4-39：在 X Window 下看中文版的 man page

不过千万不要太高兴，以为这样只要看中文的 man page 就可以了。笔者以一个很简单的数据告诉读者（如图 4-40 所示），中文版只要当作为展示中的文件就可以了，因为在中文的 man page 目录中，全部（Fedora8 为例） 也只有 10 个 man page 文件可供用户查询（有些系统甚至一个都没有）；反观一般英文版的 man page 数量则高达近 10 000，等于几乎所有命令都查询的到，还是依赖英文版的 man page 会比较安心，中文版可能要等数量再多一点才比较实用。

```
[root@LinuxTree share]# ls man/zh_TW/
man1  man8
[root@LinuxTree share]# find man/zh_TW/ -type f|wc -l
10
[root@LinuxTree share]# find man/man[1-8] -type f|wc -l
9719
[root@LinuxTree share]#
```

图 4-40：中文与英文 man page 数量的比较

4.5.10 /usr/src

虽然不是一个必要目录，但少了它有时却非常麻烦，因为这里主要储存 kernel source code 的文件。何时须要用到 source code？这是一个问题，因为如果不需要，当然这个目录就不用考虑太多，但如果用户在安装新模块或需要一些 kernel 新支持的功能时，万万不能没有这个

目录。

接下来所提到的目录，每一套操作系统版本可能在【/usr/src】下储存不同的子目录，但都是大同小异。一般 Redhat 系统在/usr/src 下会分为两大子目录。

◆ kernels

有时在【/usr/src】下会有一个“linux”目录，其实就是指在【/usr/src/kernels】中，其中默认启动用 kernel 版本的 source code，所以这句话有以下几个意思。

1. 这个 linux 目录可能只是一个链接文件（link，如图 4-41 所示），很多程序在安装或配置时，默认会指定“/usr/src/linux”为系统的 kernel 所在位置，为了让这些软件在运作时没有问题，“通常”会产生一个名称为“linux”的 link 文件，让这些软件找到所要的 kernel source code。

   在 RHEL 中，将 kernel 的 source code 直接再放到【/usr/src/kernels】目录下的【kernel version】目录中，所以一样都是 kernel source code，但目录却有差别，这在有些软件安装时会发生问题。

```
linux:/usr/src # pwd
/usr/src
linux:/usr/src # ls
.  ..  dicts  kernel-modules  linux  linux-2.6.13-15  linux-2.6.13-15-obj  linux-obj  packages
linux:/usr/src # ls -l linux
lrwxrwxrwx  1 root root 15 Jan 24 10:44 linux -> linux-2.6.13-15
linux:/usr/src #
```

图 4-41：SuSE 下的/usr/src 目录清单范例

2. 一个操作系统不只能拥有一个 kernel 版本（如图 4-42 所示），默认会配置用某一个 kernel 为启动 kernel，因为当系统中的内存大小（4GB、64GB 都是一个瓶颈）、功能（如虚拟操作系统）、测试（按照测试之目的不同）的条件不同时，就须要用不一样的 kernel 来做启动 kernel。

```
[root@PXE-ManualInstall kernels]# pwd
/usr/src/kernels
[root@PXE-ManualInstall kernels]# ls
2.6.9-34.EL-hugemem-i686  2.6.9-34.EL-i686  2.6.9-34.EL-smp-i686
[root@PXE-ManualInstall kernels]#
```

图 4-42：Redhat 系统中常见的默认 kernel 种类

目录虽不复杂，但须要注意的问题是，当用户有机会使用不同的 kernel 时，往往自己会忘记要将安装软件或模块所指定的 kernel source code 路径，切换到系统目前所使用的 kernel 版本目录，也就是 kernel 版本与所安装的软件所需之 kernel 版本不一致，这样就一定会造成执行时的错误。

◆ redhat

本目录大部分的使用时机是在使用 Source RPM 文件，一般的 RPM 文件安装完会直接将所需要的文件放入该目录中，用户无需额外做任何的动作或调整，但如果得到的是 Source RPM 文件，情况就不太一样。

利用 Source RPM 安装文件（如图 4-43 所示），一开始所安装到的路径就不会是一般的目录，而是会将所有文件先放到【/usr/src/redhat】目录下（当然还是因系统版本而异，如在 SuSE 系统中目录名称为【/usr/src/packages】），用户必须自行到该目录中，再使用 RPM 的组件另行安装，所以执行步骤就比较麻烦，但好处是没有版本限制。

```
[root@LinuxTree ~]# find /usr/src/redhat/ -type f
[root@LinuxTree ~]# rpm -ivh tg3-3.81c.src.rpm
   1:brcm-tg3-smp           ########################################### [100%]
[root@LinuxTree ~]# find /usr/src/redhat/ -type f
/usr/src/redhat/SPECS/tg3.spec
/usr/src/redhat/SOURCES/tg3-3.81c.tar.bz2
[root@LinuxTree ~]#
```

安装完Source RPM文件后所产生出的文件

图 4-43：利用 Source RPM 安装文件的过程之一

4.6 临时目录【/tmp】

/tmp 目录是一个开放的空间，可供该系统中所有用户暂时使用，所以，有一些特别的机制是必要的。/tmp 目录的属性和一般的目录差异在于其目录权限的配置（如图 4-44 所示），如果各位读者仔细看，其最后一位（用户组的执行配置）不是“x”，而是“t”。

这牵涉到文件权限属性的运作方式，一般权限除了“r”（读）、“w”（写）、“x”（执行）外，还有另外三种：

- set user id：这是针对用户本身的权限配置，可开放该文件给其他用户使用。
- set group id：这是针对用户组本身的权限配置，可开放该文件给其他同用户组的用户使用。
- sticky bit：这权限开启代表开放权限给“所有人”使用，所以，/tmp 目录就是要将该权限开放给所有系统中的用户。

```
[root@LinuxTree ~]# ls -ld /tmp/
drwxrwxrwt 17 root root 4096 2008-02-22 11:30 /tmp/
[root@LinuxTree ~]#
```

图 4-44：/tmp 目录的权限配置

至于其他详细的权限用法，可参考网络或其他书的介绍，在此不再赘述。但因为权限的因素，系统在设计时，/tmp 目录中所默认存在的目录及文件，大都是为了要让用户共享的信息（如很多 X Window 在用的文件），不过，并不都是“默认存在的”，有部分信息是当该软件使用时才会产生，也正因为如此，会选择以一个任何人都可在其下建立目录或文件的位置做暂存区。在/tmp 下其实有非常多临时或不同的软件所存放的数据，在此仅将几个常会用到的目录或文件列出，以供参考。

4.6.1 /tmp/.font-unix

以往的 X Window 配置文件中，都会有一个“unix/:7100”的字体路径的配置，若在启动时无法找到该路径，会造成 X Window 的无法使用。这个“7100”其实就存在于/tmp/.font-unix 的目录中（如图 4-45 所示），以前如果不小心将/tmp 目录清空（因为该目录是以“.”开头，所以不会注意到），X Window 就等于无法使用（因为没有字体连接）。不过，现在较新的 X Window 版本看来已经解决这个问题，因为即使将这个目录整个删除，虽然还是会有错误信息，但仍可以进入图形画面。

```
[root@LinuxTree tmp]# ls .font-unix/
fs7100
[root@LinuxTree tmp]#
```

图 4-45：.font-unix 目录中只有一个文件

4.6.2 /tmp/gconfd-juergen

其实这个目录名称应该叫做（gconfd-“User”），因为每一个用户都会有自己的目录。此目录一般都是空的，因为少有用户会将 X Window 的登录模式配置为“自动登录”，而这个目录所存放的是一个登录密钥，如图 4-46 所示。

```
[root@LinuxTree gconfd-juergen]# find
.
./lock
./lock/ior
[root@LinuxTree gconfd-juergen]#
```

图 4-46：gconfd 下的密钥文件

当用户刚启动或利用【init 5】要进入 X Window 时，登录程序（gdm）如果发现存在这一文件（当然也要有 gdmsetup 命令配置自动登录选项），就可以通过这个文件省略登录的步骤。当然，如果是用【startx】的命令，原本就不需要登录画面，因为这和【init 5】的登录步骤本身就有差异（详细请参考笔者的另一著作《Linux 操作系统之奥秘》），所以这个文件就更不会用到

了。如果有些程序须要自动登录或密钥控管，可能就会使用到/tmp 下的【gconfd-User】目录。

4.6.3 /tmp/.ICE-unix

在 X Window 下，不论是哪一个用户，登录到 X Window 的画面后，X Window 就会自动在这目录下产生一个专属的 session 文件（如图 4-47 所示），而这些 session 文件代表以下几件事情。

1. 一个登录者会有一个专属的 session 文件，请注意是登录者而不是账号，因为同一个账号可以在好多个 console 中执行，意思是同时可以有好几个 X Window 在系统下运作，所以有可能同一个账号却使用好几个 session 文件。
2. 当用户“注销”时会将该 session 文件删除，但这也代表着，如果不照正常程序注销，该文件将会变成一个垃圾文件存留在那边（如图 4-47 中有三个是 juergen 的 session 文件，只有一个是目前在使用中的）。所谓不正常注销，包含如【Ctrl+Alt+Backspace】或【init 3 命令】等，如果不希望有这些文件残留在计算机中，就请按照规定“注销”，或者重新启动。
3. 原本的 session 如果在 X Window 使用中被删除，界面还是可以使用，但会产生一些问题，比如，Terminal（仿真控制台软件）开启时就会变得很慢，问题应该就是出在找不到原本的 session 文件。

```
[root@LinuxTree .ICE-unix]# ls -l
total 0
srwxrwxrwx 1 juergen juergen 0 2008-02-22 14:55 10109
srwxrwxrwx 1 juergen juergen 0 2008-02-22 14:55 10351
srwxrwxrwx 1 root    root    0 2008-02-22 14:56 10589
srwxrwxrwx 1 juergen juergen 0 2008-02-22 14:53 9606
[root@LinuxTree .ICE-unix]#
```

图 4-47：.ICE-unix 下的 session 文件

总结

这些应用程序的目录，大部分都是系统厂商自行决定该如何使用，因为对系统使用的影响不会太大（反正有命令的默认路径可做配置），所以和正规的路径往往会有一些出入，不过，最主要的大方向还是不会变的。

熟悉这些应用程序目录，对使用有很大的帮助，因为用户在安装很多软件时，对软件所安装的目标路径不是很清楚，要自行配置或用软件的默认值也不知该如何取舍，对这些目录比较熟悉后，相信对安装时的目录配置会清楚很多。

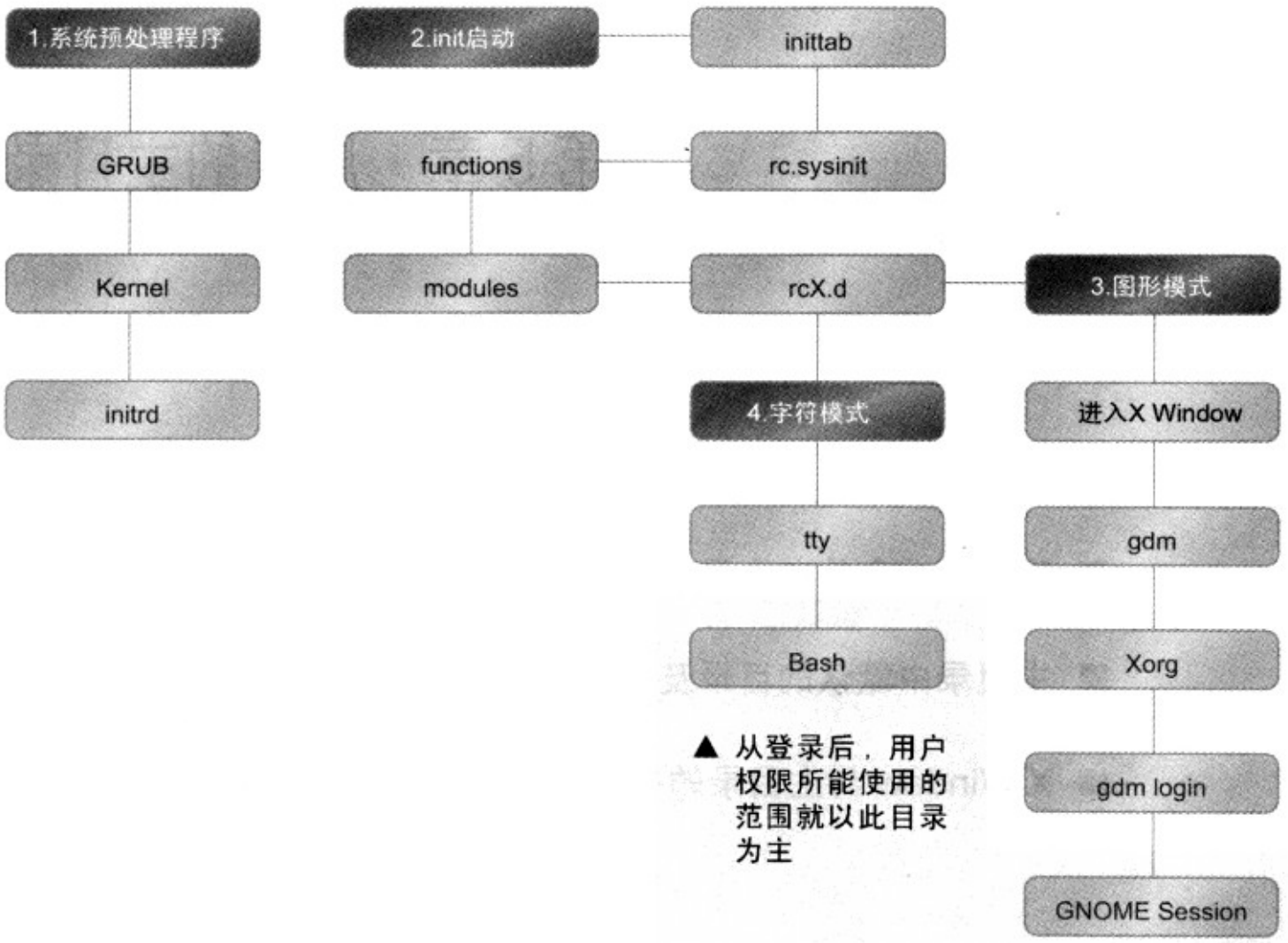
第5章　用户的主目录

本章学习重点

- 哪些目录会放在主目录中
- 主目录中默认的目录及文件从何而来
- X Window 与主目录的关系

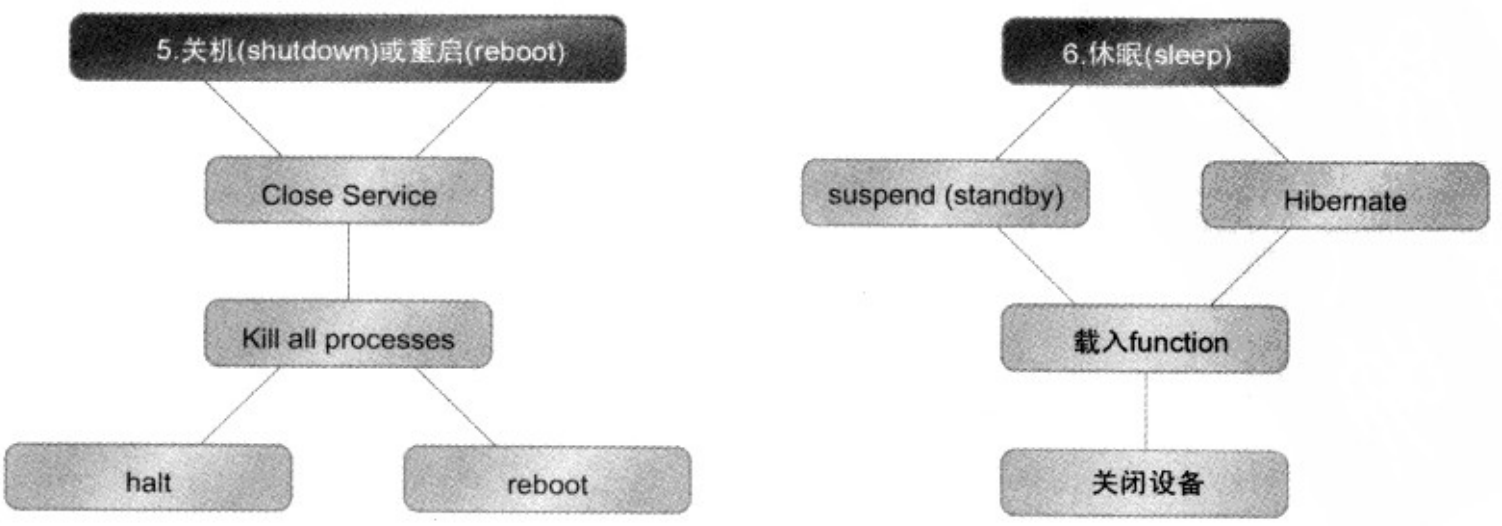
系统流程与章节内容对照图

- 启动流程



▲ 从登录后，用户权限所能使用的范围就以此目录为主

- 关机流程



主目录是从用户一开始登录系统后（如图 5-1 所示），即可被允许在该目录下做任何用户想做的事的目录，因为这一目录专属于该用户，一般用户因为目录权限的关系，默认无法观看或进入（一般的默认权限）其他人的主目录，但这也是确保每位用户隐私的唯一办法。

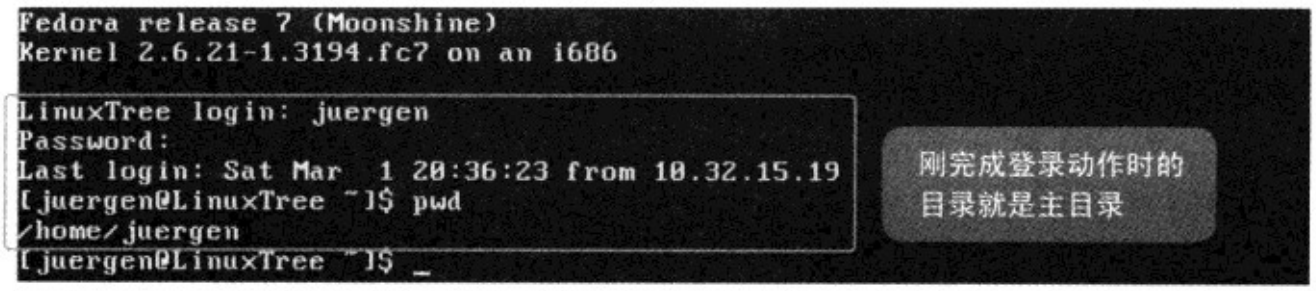


图 5-1：刚完成登录时的用户目录

将所有用户的主目录区隔开来是一件很重要的事，就好比现实生活中每一个家都有一道门，若这道门被打开，就会变成无人防护的状态，相对里面的安全自然大受威胁。

如图 5-2 所示，一般用户主目录的权限（上方的框选处）在一开始建立时，就已经被限制为只有该用户可以【读 / 写 / 执行】，其他用户连“读”的权利都没有，等于是将个人用户的主目录完全保护着。

至于管理员的主目录（下方的框选处），则是被限制为除个人使用外，尚可供同用户组的用户【读 / 执行】，主要是因为管理员同时会有许多同用户组（root 群组）的用户协助管理，所以可能须要使用到存在于管理员主目录（/root）中的文件，因此，除了写入的功能外，读与执行都在可容许的范围内。

这些权限的配置都是系统的默认值，用户当然可以自行调整，但就像先前所提及的，其权限好比是家里的门锁，而且是一个比较严谨的方式，如果自行调整成比较宽松的条件，很有可能会“引狼入室”。

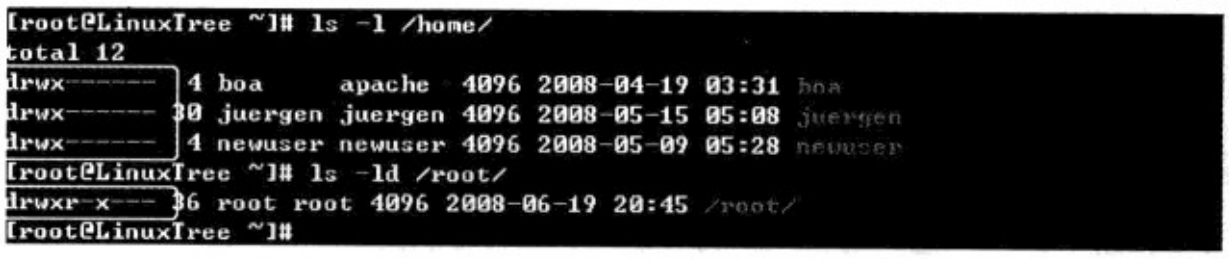


图 5-2：主目录的权限

Linux 默认的用户主目录，是在【/home】底下的“用户名称”目录中，本章会以“juergen”为用户名称的示例来说明。所有用户中，唯一主目录默认不放在/home 下的是系统的管理员“root”，因为必须和一般的用户有所区别，如文件大小、使用权限等可能的一般用户限

制，所以将 root 的主目录放在/root 的目录中，不过，因为/root 的目录结构和一般的用户大同小异（只是用户不一样），在此不多做介绍，接下来只以一般用户的主目录进行说明。

5.1 /home/juergen/基本文件

本章的【UserName】都将使用【juergen】的名字当作示例用户名称，在开始进行/home/juergen 目录下的子目录之前，先说明“/home/juergen”的产生原因。

当管理员新增用户时，系统会自动为该用户新建一个主目录，取名的方式就是以新用户的名称作为目录名称（如图 5-3 所示），但这是默认值，也就是说，当然可以不一样，稍后的内容会再介绍。

```
[root@LinuxTree ~]# ls /home/
[root@LinuxTree ~]# useradd juergen
[root@LinuxTree ~]# ls /home/
juergen
[root@LinuxTree ~]#
```

图 5-3：系统新增用户时自动在/home 下产生对应的主目录

当产生用户的目录时，系统会顺便将一些默认要放入主目录中的文件一并产生，重点来了！这些默认的目录或文件是从哪来的？配置文件或实际有的目录呢？这个问题很重要，因为如果可以知道这些源文件放在哪里，就能帮助管理员在建立用户账号时，设置限制或开放哪些文件目录给用户。

实际上，要存放哪些文件在主目录中，全看【/etc/skel】目录中的配置（如图 5-4 所示），系统会在建立用户主目录的同时，将【/etc/skel】目录下的内容复制一份到该用户的主目录中，不过，当然复制到该用户目录中的所有文件及目录，都会将其所有者与用户组自动替换为新的用户，有关【/etc/skel】的说明也可以参考本书“第 6.1.3 节：系统目录”【/etc/skel】的内容，会有对这个目录的说明。

```
[root@LinuxTree ~]# ls /home/juergen/ -a
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .kde  .xemacs  .zshrc
[root@LinuxTree ~]# ls /etc/skel/ -a
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .kde  .xemacs  .zshrc
[root@LinuxTree ~]#
```

图 5-4：/home/juergen 和/etc/skel 目录的比较

刚有提过，其实/home/juergen 是一个系统默认的目录，也就是说，用户也能不照默认的方式来摆放主目录的位置，甚至可以自由调动或配置主目录的放法（比如，一个部门一个“home”，像/home1、/home2 等），这都是很容易做到的，毕竟 Linux 是一个完全自由的操作系统。

以一个简单的例子来说，如果要将一个用户 paul 的主目录设置在/home 之外的/home1 目录，只须要按照以下几个步骤就可以完成（如图 5-5 所示）。

Step 1 建立/home1。

Step 2 新增 paul 用户时，另外增加“指定主目录”的参数。

Step 3 完成，是不是很简单呢！

```
[root@LinuxTree ~]# ls -ld /home*
drwxr-xr-x 3 root root 4096 2008-03-02 01:10 /home
drwxr-xr-x 2 root root 4096 2008-03-02 01:21 /home1
drwxr-xr-x 2 root root 4096 2008-03-02 01:20 /home2
[root@LinuxTree ~]# useradd paul -d /home1/paul
[root@LinuxTree ~]# grep paul /etc/passwd
paul:x:502:502::/home1/paul:/bin/bash
[root@LinuxTree ~]#
```

图 5-5：将主目录移出/home 外

其实，不一定要在新增用户时指定主目录，也可以在建立后，通过/etc/passwd 文件的格式来做修改（如图 5-6 所示），观察系统管理员的做法，市面上也有许多帮助管理员建立用户的软件，可一并做配置。重要的是，用户的新增或删除也是需要事前先规划好的，在建立所有系统的用户前（如果是企业内部用户，可能用户数目会很大）要先考虑好，有哪些目录或文件是要每一位用户都需拥有的，不然等用户都建立完成后再考虑或修改，会比较麻烦。

```
[root@LinuxTree ~]# tail -3 /etc/passwd
juergen:x:501:501::/home/juergen:/bin/bash
boa:x:502:48::/home/boa:/bin/bash
newuser:x:503:503::/home/newuser:/bin/bash
[root@LinuxTree ~]#
```

主目录的路径可在这更改

图 5-6：passwd 文件中有关主目录的部分

刚刚是系统默认在主目录下所产生的目录或文件，但在接下来介绍的目录中，有很多不在图 5-6 的目录清单中，这是因为很多常用到的软件或 X Window，都需要在用户的主目录下，自行产生一些配置文件或存放数据的目录，所以接下来的章节，不一定和每一位读者的环境条件一样。

5.1.1 .bashrc 及.bash_profile

.bashrc 及.bash_profile 这两个文件是用户登录时所使用的 shell 环境（默认大多为 bash）的配置文件，以 Fedora 为例，当用户登录时，会以一连串的动作完成登录程序（如图 5-7 所示），相对来说，这两个文件所扮演的角色就变得非常重要了。

因为在一般用户的登录过程中，无法加入一些个人的配置值（root 或其他管理员可以利用很多额外的系统文件完成自动化的操作），大部分都是系统默认值，如果希望在登录时自动完成一些程序，就可以利用这两个文件。

在一些 Linux 版本中默认或许没有这两个文件，但用户可以自行产生，因为其实没有这两个文件，用户仍然可以正常运作（只是会因为少一些变量或别名使用上不太习惯），所以有些系统在不必要的情况下就不会提供现成的文件。

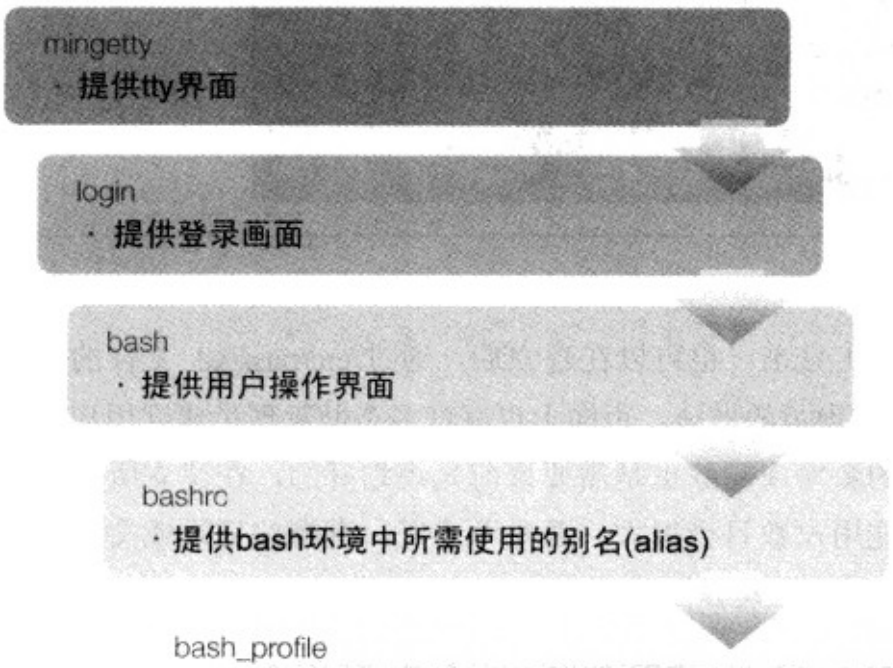


图 5-7：用户从看到登录画面到登录后的部分过程

【.bashrc】与【.bash_profile】这两个文件的方便之处，在于提供用户一个弹性的空间，让所有用户都可以自定义一些流程，使系统在登录后可自动运作（如图 5-8 所示）。图中先从“juergen”登录，登录后紧接着就直接显示两行信息（“bashrc file”与“profile file”）。

这两行其实就是笔者在【.bashrc】与【.bash_profile】中所注记的，目的是为了表现出两件事：

- 【.bashrc】与【.bash_profile】两个文件对系统所做的自动化，用户无须再多做额外的动作，系统直接会加载新加入的执行动作。
- 【.bashrc】与【.bash_profile】所执行的顺序：通过这个结果不难发现，一定是先执行完【.bashrc】后，才会再执行【.bash_profile】。

```
Fedora release 7 (Moonshine)
Kernel 2.6.21-1.3194.fc7 on an i686

LinuxTree login: juergen
Password:
Last login: Fri Jun 20 01:26:18 on tty2
bashrc file
profile file
[juergen@LinuxTree ~]$ tail -1 .bashrc
echo "bashrc file"
[juergen@LinuxTree ~]$ tail -1 .bash_profile
echo "profile file"
[juergen@LinuxTree ~]$ _
```

图 5-8：用户自定义登录的流程

5.1.2 .bash_history

主要记录用户曾经输入过的所有命令，当然不可能是无上限的记录，最多 1 000 条（主要以用户的变量配置为主，如图 5-9 所示），为了让用户在输入过某些指令后，后面可以有记录追查之前的动作（如用户登录后，可以用 ↑ 键，追查出之前曾经使用过的命令）。

这一份“命令记录”的文件，在大部分的操作系统中，都以【.bash_history】为文件名称，不过其实这是可以自行更改的，因为这个文件的目的仅在于供用户使用上的方便，并非有所限制。

因此，从图 5-9 中可以看到，其实用户只需要将以下的【HISTFILE】变量值改变（可直接在 shell 下输入改变参数的命令“HISTFILE=/home/juergen/.bash_testfile”），就可以达到更改文件名的目的（文件不用建立，系统会自动建立），唯一要记的就是让变量每次启动都更改即可。

```
[juergen@LinuxTree ~]$ env | grep -i hist
HISTSIZE=1000
[juergen@LinuxTree ~]$ set | grep history
HISTFILE=/home/juergen/.bash_history
SHELLOPTS=braceexpand:emacs:hashall:histexpand:history:interactive-comments:moni
tor
[juergen@LinuxTree ~]$ _
```

history的文件名也可以被变更的

图 5-9：变量和 history 文件之间的关系

【.bash_history】并不是一个实时的记录文件，也就是说，在用户登录后，并不会将所有该次登录后的命令一并记录，而是在注销时，由系统一次将所有该次登录所使用的命令全部写入【.bash_history】文件中，所以，当次输入的命令都要等该次登录注销后才能够在文件中看到。

因此，【.bash_history】文件真的只是一个历史文件，并非可以实时检查该次登录所输入的命令，如果有须要查询该次登录所输入的命令，除了按【 ↑ 】的按键外，还可以通过【history】命令做查询（如图 5-10 所示）。

```
[juergen@LinuxTree ~]$ history | tail -5
  110  ls /etc/
  111  ps ax
  112  set
  113  env
  114  history | tail -5
[juergen@LinuxTree ~]$ abcde
-bash: abcde: command not found
[juergen@LinuxTree ~]$ history | tail -5
  112  set
  113  env
  114  history | tail -5
  115  abcde
  116  history | tail -5
[juergen@LinuxTree ~]$
```

图 5-10：使用 history 命令查询当次登录所使用过的命令

5.1.3 .bash_logout

每一个用户在使用完系统后，都要做一次注销的动作（不管是使用【exit】或【logout】都一样），既然有动作就可以控制，这是 Linux 不变的法则。对注销而言，可以利用【.bash_logout】文件来做自动化。

也就是说，当用户注销的同时，系统会自动执行【.bash_logout】文件（如图 5-11 所示，默认只会做清除窗口的动作），如果管理员须要记录用户注销的一些额外记录、动作或其他信息，也可以利用这一个机制去完成。

```
[juergen@LinuxTree ~]$ cat .bash_logout
# ~/.bash_logout

/usr/bin/clear
[juergen@LinuxTree ~]$
```

可以在其中加入注销所需的动作

图 5-11：用户注销时所使用的文件

5.1.4 public_html

这个目录其实不算是系统默认的目录，而是网页服务器组件（默认大部分都为 Apache）所默认的用户网页存放的目录名称，所以当管理员在建立用户时，一般会先帮用户建立出【public_html】的目录（当然不会一个一个建立，一样可以利用/etc/skel 的目录做预先的目录和文件规划）。

刚刚提到这个目录名称是网页服务器所需要的，自然此目录的名称也是按照服务器软件的配置而定（如图 5-12 所示），只要可以让网页服务器的用户目录设定与用户主目录中的目录名称一致（其他像权限、网页名称等配置暂不考虑，因为那是建站的事情），就可以将用户的网页

呈现在大家面前。

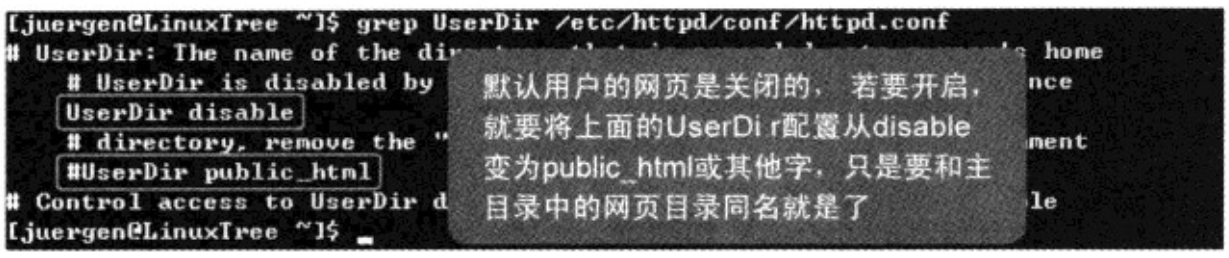


图 5-12：Apache 中有关用户网页目录的配置

既然提到网页的目录，就顺带提醒一下读者，若希望在 Linux 下的主目录中放置一些个人的网站或网页，最少要先确定“主目录”的权限是否打开，否则是没有人有权限读取主目录中的文件的。

请读者回想一开始所提及的“主目录”，如【/home/juergen】，为了目录中的安全性，默认为只提供个人使用，所以基本的权限为“700”，只开放个人读取。但当要提供网页或文件给其他的用户读取时，就必须将其他用户的权限也考虑进来，所以权限最少要【705】（如图 5-13 所示，增加其他人读与执行的权限），才可以将主目录中的网页及文件共享出去。

```
[juergen@LinuxTree ~]$ ls -ld /home/juergen/
drwx---r-x 32 juergen juergen 4096 2008-06-20 02:44
[juergen@LinuxTree ~]$
```

图 5-13：开放主目录浏览网页的权限

5.2 /home/juergen/额外文件

除了在“第 5.1 节：/home/juergen/基本文件”中所提到的目录及文件外，还有很多因软件而产生的目录及文件都会被放在主目录下，因为每一个用户都有自己的配置（如网页软件），所以主目录是一个最好的存放地方。不过，因为是某些软件所产生的，所以每一套软件都有其自己的存放方式，每一个操作系统也有可能会有不同的表现方式，这里只能做一个参考，而且大部分会看到的目录或文件，都是 X Window 下的“产物”，换句话说，若没有使用该用户的账号登录过 X Window，是不会产生这些“产物”的，当你看不到这些目录或文件时，请先尝试登录 X Window，让系统自动产生应有的目录或文件。

除了 X Window 外，其他的软件也如此，都必须先执行过该软件，经过该软件判断在第一次启动后，才会帮用户在正确的主目录下建立相对应的目录或文件夹，以下将一些常见的目录列出并说明。

5.2.1 X Window 配置

X Window 在启动时，除了会参考主要的【/etc/xorg.conf】X Window 配置文件之外，还有一些相关用户个人配置的信息在主目录中，包括所有在 X Window 下启动的程序，都有一份配置在里面。

不过，因为在主目录下的目录规划并没有特别的规定，唯一的要求就是所有的软件配置，在用户主目录下都必须存放在“.”开头的“目录”中，换句话说，在主目录中不会看到任何一个非“.”开头的应用程序配置文件，这也是为什么每一套软件的文件放置方式都可能不一样的原因。

举例来说，常会用到的 gedit 程序（就是文字编辑器）及 FireFox 网页浏览程序，两者配置文件的位置就天差地远（如图 5-14 所示），但因为 X Window 下的软件正常都可直接在 X Window 下做配置，所以这些目录的重要性相对就比较低，其中大部分结果都不会影响系统的运作，都是偏向软件的配置。

```
[juergen@LinuxTree ~]$ ls .gconf/apps/
%gconf.xml  gedit-2  nautilus  panel  puplet
[juergen@LinuxTree ~]$ ls .mozilla/
firefox
[juergen@LinuxTree ~]$ _
```

图 5-14：应用程序在主目录中的配置文件存放位置

当然在主目录中，除了这两个应用程序的目录外，还有许多目录（如图 5-15 所示，都以“.”为开头的目录）都是应用程序所存放配置文件或文件的地方，如 GNOME 及 KDE 也都有其专属的目录（如【.gnome2】及【.kde】）；但这些目录都只是针对/home/juergen 所设计的，也就是说，在别的用户主目录中，不一定会有一样的目录产生，因为之前提过，这些绝大部分都是在“第一次”启动该软件时被建立出来的，如果没有执行过，当然不会有这些目录的产生。

```
[juergen@LinuxTree ~]$ ls -d .*
.               .config   .gnome2              .kde                .redhat
..              .dmrc     .gnome2_private      .lesshst            .Trash
.bash_history   .emacs    .gstreamer-0.10      .metacity           .viminfo
.bash_logout    .gconf    .gtk-bookmarks       .mozilla            .xemacs
.bash_profile   .gconfd   .gtkrc-1.2-gnome2    .nautilus           .xsession-errors
.bashrc         .gnome    .ICEauthority        .openoffice.org2.0  .zshrc
[juergen@LinuxTree ~]$ _
```

图 5-15：主目录中应用程序相关目录

在此列出几个在主目录中，使用 X Window 时很实用的几个文件。

◆ ~/.Xclients

当用户登录 X Window 时，某些文件会被读取，牵涉到的文件非常多，主目录自然也不在话下，因为用不同的账号登录，就会有不一样的结果（如桌面、目录等，这和 Windows 的概念是一样的），而这是其中一个可利用的配置文件实例，但主目录默认不一定会有。

.Xclients 是一个登录 X Window 的脚本文件，也就是说，X Window 是按照这个文件逐一执行的，因此，需要比较细的逐步调配，编辑这个文件（如图 5-16 所示）有一些小细节须要注意。

- .Xclients 的文件权限要有执行（“x”）的权限。
- 每一个执行的步骤都须要加“&”，将执行过程放到后台执行，不然会影响 X Window 的执行过程。
- 最后面要加入“gnome-session”，因为前面的执行完成后，须要告诉 X Window 开始加载桌面的环境（由 gnome-session 起头），但只有这一行最后面不要加“&”符号。

```
[juergen@LinuxTree ~]$ ls -l .Xclients
-rwxrwxr-x 1 juergen juergen 31 2008-06-20 04:46 .Xclients
[juergen@LinuxTree ~]$ cat .Xclients
gnome-terminal &
gnome-session
[juergen@LinuxTree ~]$
```

脚本文件的内容

图 5-16：可帮助 X Window 登录时自动执行的文件

这些小细节若没有注意到，很容易会卡在某一个阶段，造成 X Window 的启动失败或停止。比如，图片文件中的【gnome-terminal】命令，刚刚提到在最后面一定要加“&”符号，如果把“&”符号移掉，会发现 X Window 将直接停在【gnome-terminal】环境中（如图 5-17 所示），如果要进入 X Window 的桌面环境，就必须离开（exit）【gnome-terminal】，才会执行脚本文件中接下来的【gnome-session】。

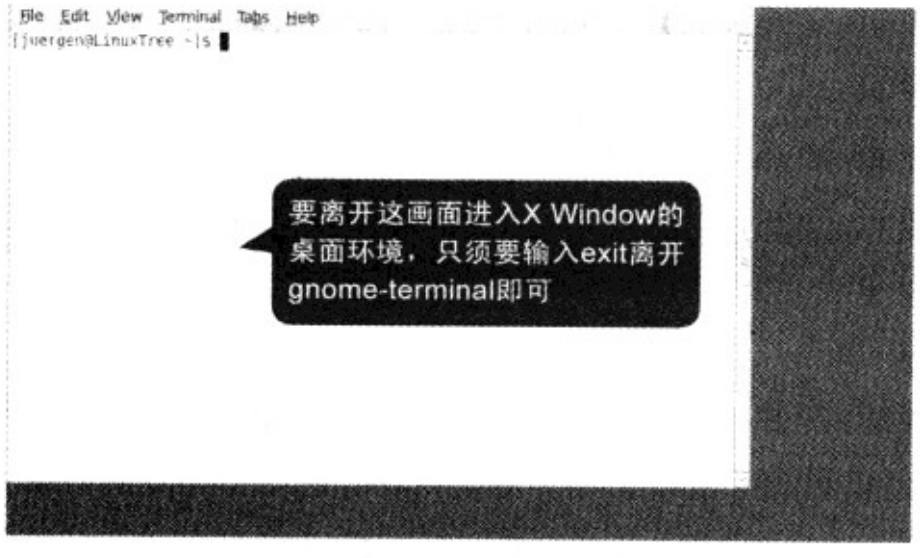


图 5-17：因执行 gnome-terminal 而停住的 X Window 画面

◆ ~/.gnome2/session

接续上一个.Xclients 的话题，刚刚最后执行的一条命令是 gnome-session，而 gnome-session 主要是负责启动 GNOME 桌面环境及软件的加载，所以有几个文件会和软件的启动有关，读者要先清楚文件的加载顺序，如图 5-18 所示。

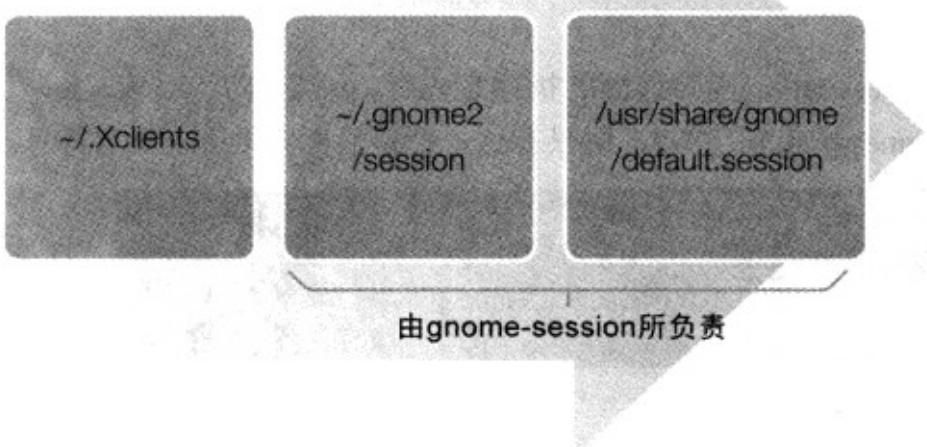


图 5-18：X Window 启动时的部分顺序

由这个顺序可以得知，当前一个.Xclients 执行完毕，进入 gnome-session 时，可以利用主目录中【~/.gnome2/session】或【/usr/share/gnome/default.session】这两个文件做进入画面时的动作调整。

在主目录中默认应该是没有【~/.gnome2/session】文件的，因为大部分所须执行的软件都按照默认值执行，因此，有以下两种方式。

- 直接修改【/usr/share/gnome/default.session】文件，但这样是所有用户都参照此配置。
- 自行建立【~/.gnome2/session】文件，这也是本节所要强调的，因为也是在主目录中的配置文件，可以依不同用户，也就是个人的需求做调整，弹性比较大。

默认主目录下的 session 文件内容（如图 5-19 所示），比刚刚所看到的.Xclients 复杂，因为其中加入了优先权的概念，但其实调配十分简单，若没有很特殊的需求，只要将所须执行的文件加在最下面，优先权数字用高一点（也就是比较后面执行）就可以了。

不过，session 文件和.Xclients 比起来还有一个小小的好处，就是文件权限比较弱，因为它是一个被读取的文件，不像.Xclients 需要有执行的权限，一不小心就会忘记赋予所需的权限。

```
[juergen@LinuxTree .gnome2]$ cat session
[Default]
num_clients=6
0,id=default0
0,Priority=60
0,RestartCommand=pam-panel-icon --sm-client-id default0
1,id=default1
1,Priority=10
1,RestartCommand=gnome-wm --default-wm gnome-wm --sm-client-id default1
2,id=default2
2,Priority=40
2,RestartCommand=gnome-panel --sm-client-id default2
3,id=default3
3,Priority=40
3,RestartCommand=nautilus --no-default-window --sm-client-id default3
4,id=default4
4,Priority=40
4,RestartCommand=gnome-volume-manager --sm-client-id default4
5,id=default5
5,Priority=50
5,RestartCommand=gnome-terminal --sm-client-id default5
[juergen@LinuxTree .gnome2]$ ls -l session
-rw-r--r-- 1 juergen juergen 561 2008-06-20 07:28 session
[juergen@LinuxTree .gnome2]$
```

这一个软件的数目是需要和真正在这个文件中所执行的文件数目一致

各软件所执行的优先级

图 5-19：session 文件的示例

◆ ~/.xinitrc

基本上，这个文件的作用在某些地方和之前提到的【.Xclients】文件是一样的，都是 X Window 的脚本文件，可供用户定义要执行哪些 Client 软件，如【gnometerminal】控制台程序。

不过比较不一样的是，这个文件是【startx】与【gdm】（GNOME 所使用的 X Window 启动工具程序）可共享的文件，但【.Xclients】是只有 GNOME 所提供的 gdm 管理程序（正确的说法应是 xdm，因为有多种 Desktop Manager）所使用的，所以在可使用的范围上比较不一样，需要特别注意。

【.xinitrc】文件与上述两个文件的差异主要来自于启动 X Window 的方式，所以可以启动 X Window 的可配置文件，如图 5-20 所示。

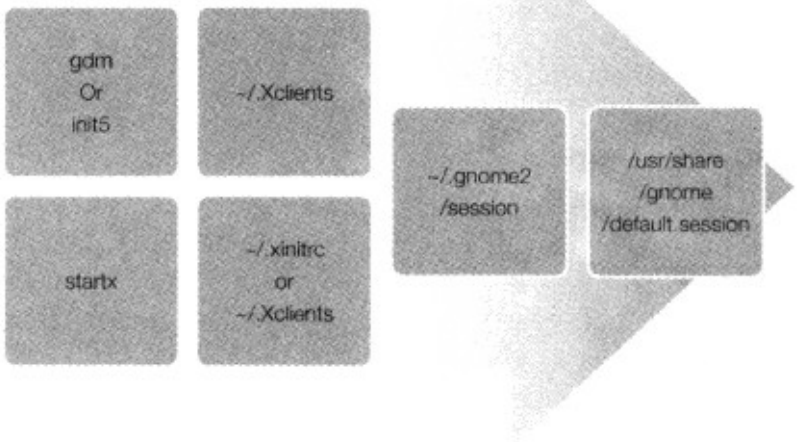


图 5-20：再度调整过的 X Window 文件顺序

◆ ~/.xsession

这个文件就不再赘述，因为其作用和【.Xclients】是一模一样的，差别只在用户习惯用哪一个文件做配置而已，不论用哪一个都可以让用户在 X Window 中执行定制化的程序。

看到这几个文件，虽然数目不是很多，但在 X Window 与启动的顺序上，一定还会有一些混淆或不够清楚的地方，故在本节的最后，笔者还是将所有 X Window 启动的关系（详细信息请参考笔者的另一本著作《Linux 操作系统之奥秘》）与配置文件的顺序（如图 5-21 与图 5-22 所示）用图形化的关系做一个总整理，方便读者未来可能须要做配置时参考。

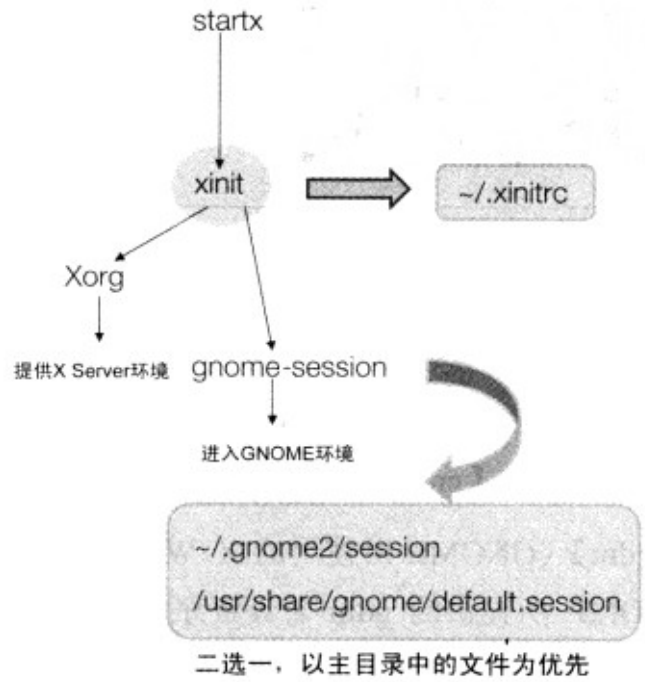


图 5-21：当系统以 startx 启动 X Window 时可使用的配置文件

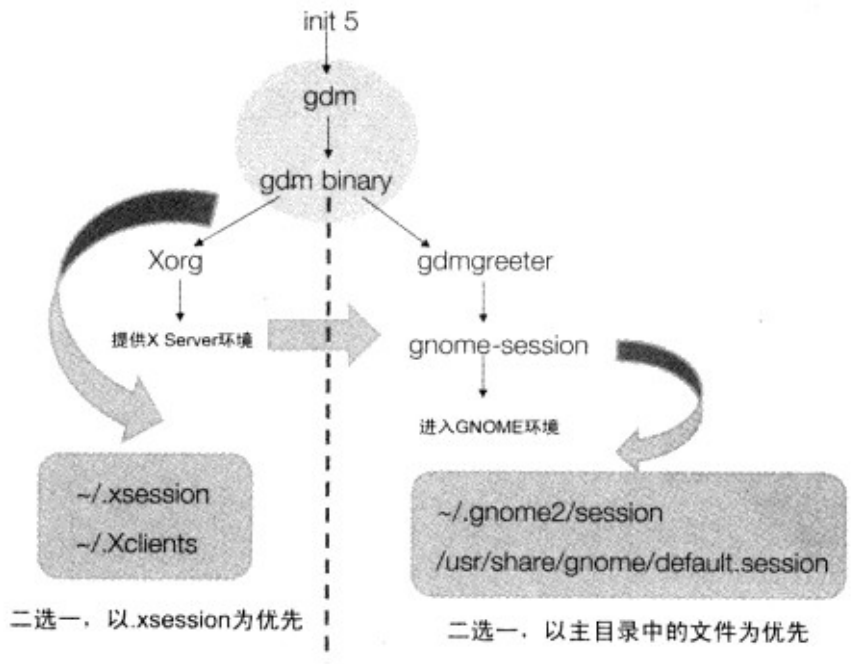


图 5-22：当系统以 init 5 启动 X Window 时可使用的配置文件

从图 5-21 与图 5-22 两张示意图，可以知道以下几件事情。

- 当启动 X Window 时，以【startx】或【init 5】命令开始，其过程差异很大，所以中间的文件过程会有差异。
- 中间所执行的文件顺序很重要，以图 5-22 为例，若【~/.xsession】脚本文件实例中没有加入【gnome-session】的执行命令，右边的所有文件及【gnomesession】管理程序就不会执行，自然就不会有 GNOME 的桌面画面，系统就会直接跳回上一步登录画面。
- 某些文件其实是重复的，如图 5-22 中的【~/.xsession】和【~/.Xclients】，只要有其中一个就可以，当两个都存在的时候，只会有一个可以执行，所以当不了解时，往往会发生以为某个文件没作用的情况。
- 【startx】与【init 5】在启动的“一开始”，最大的差异就在于【startx】会读取【~/.xinitrc】文件；而在启动过程中，因为【init 5】会使用【gdm】作用管理程序，所以多了【~/.xsession】和【~/.Xclients】两个文件可供用户做配置，因此，【startx】与【init 5】两种模式各有其方便之处。

5.2.2 X Window 文件存放目录

X Window 在启动后会自动建立一些文件存放的目录，和一般 Window 下的【我的图片】、【我的音乐】、【我的影片】或【我已接收的文件】是一样的意思，只是一个系统默认的文件目录，并没有任何硬性规定说一定要放在哪。

对照到 Linux，在 X Window 第一次登录后，便会在用户主目录下建立和 Windows 类似的默认目录，如【Documents】、【Download】、【Music】、【Pictures】、【Videos】等，这些都是在主目录下专为用户所产生的，当用户在使用这些主目录时，于这几个目录下不会有权限的问题发生，如图 5-23 所示。

```
[juergen@LinuxTree ~]$ ls
1         Documents   id_rsa.pub  Music      Public        Templates
Desktop   Download    mbox        Pictures   public_html   Videos
[juergen@LinuxTree ~]$
```

图 5-23：主目录下的默认文件目录

◆ 【Desktop】

此为这些目录中最重要的一个，因为此目录就是用户在登录 X Window 后所使用的桌面环

境。从图 5-24 中可以知道，X Window 和【Desktop】目录的关系是实时的，只要用户适当地将所须放入桌面的文件，直接存入【Desktop】目录中，即可直接在 X Window 的桌面上看到该文件。

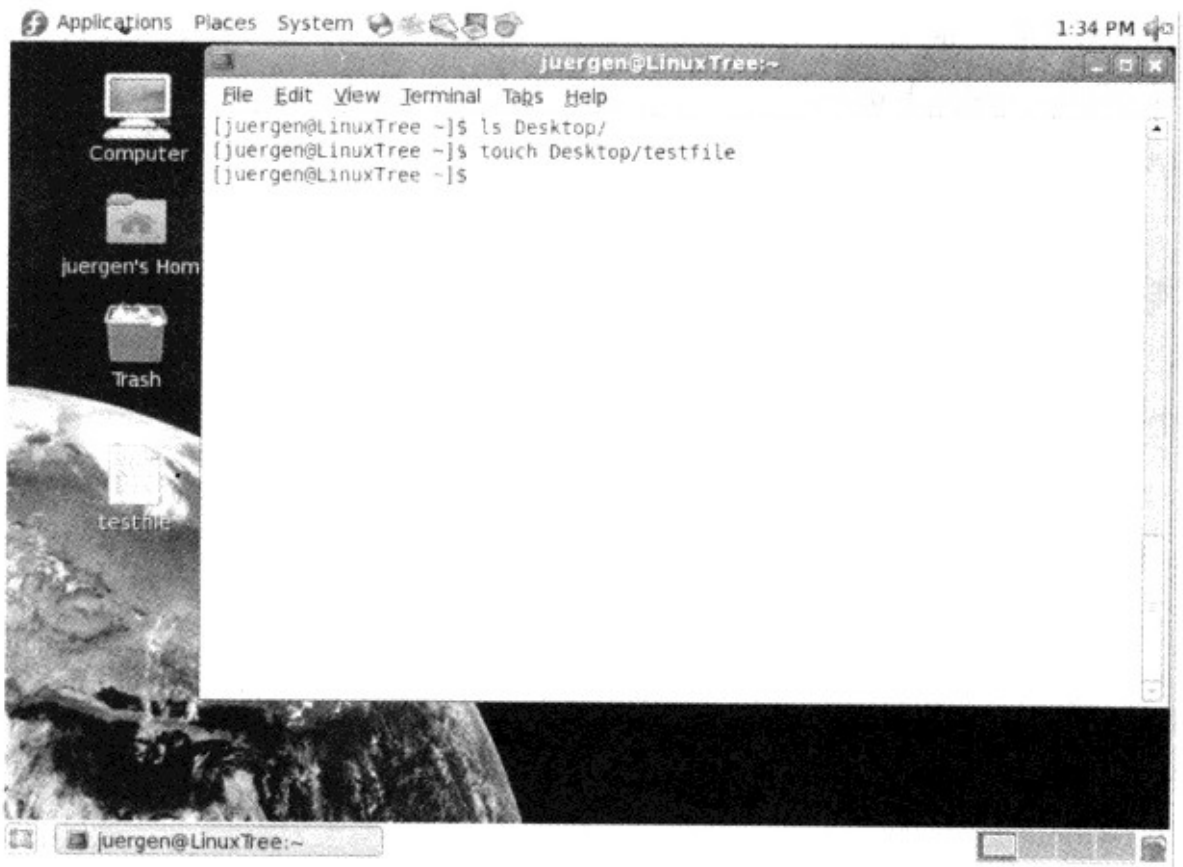


图 5-24：Desktop 目录的使用方式

◆ 【Documents】

文件目录，默认让用户存放所有用户须要编辑或储存的目录。

◆ 【Download】

下载专区，虽然这是 X Window 的默认下载目录，但很多 FTP 或相关软件默认并不会放在这个目录下。

◆ 【Music】

音乐区，可以将一些常用的音乐放在其中。

◆ 【Pictures】

照片区，所有要存放的照片都可放在其中。

◆ 【Public】

公开的目录，表示这个目录下的信息是公开的。

◆ 【Templates】

暂存区，有点像/tmp 目录，供用户本身暂存一些文件。

◆ 【Videos】

影像区，可存放电影、MV 或类似的影像文件。

这些目录主要只以目录名称分类，并没任何的实际限制，所以还要看用户的习惯与软件的配合度。因为一般登录用的是默认的英文语言，自然 X Window 就会将目录用英文命名。如果用户一开始就用中文语言登录，就会发现系统所建立的目录都是用中文（如图 5-25 所示）。不过，这有一个缺点，就是只能在 X Window 下检查，因为一般的 console 并不支持中文，当切换到 X Window 以外的环境时（包括 telnet、ssh 等的方式），就会有语言上的问题产生（除非在 shell 上安装中文组件，但其他用户通过网络登录一样会有问题）。

```
[root@LinuxTree ~]# ls
anaconda-ks.cfg  install.log.syslog  公共  影片  桌面  音樂
install.log      下載                圖片  文件  模板
[root@LinuxTree ~]#
```

图 5-25：X Window 下所看到的中文目录

另外值得一提的目录，就是【.Trash】（因为是以“.”开头的目录，所以不在图 5-25 的文件清单中）。因为这一目录就等于是 Windows 下的资源回收站，如果在 X Window 下使用 GUI 界面将某文件删除，就会被丢到这一个目录中，如果用户在没有 X Window 界面的情况下，要找删除的文件，请到这个目录中找吧！不过要注意的是，如果是在 X Window 的控制台软件下命令删除文件，是不会放到【.Trash】目录中的，因为这仍然和 console 下删除文件是一样的意思，请三思而后行。

总结

在主目录中，所有的数据都可以被该用户所管理，所以绝对不会是一个系统的默认值，但也代表在主目录下所能更改的功能是有限的。不过，因为一般用户（本章所强调的都是一般用户，因为管理员的权限太大，不受主目录局限）的权限实在是非常地小，大部分仅能在本身的

主目录中修改文件，因此，更不能放过主目录下可变更的配置，尤其是登录之后的所有步骤。

至于 X Window 及各项软件所相关的配置，基本上，可以放心交由各软件的界面所控制，毕竟这些配置或目录都是由各软件直接提供的，在更改上通过这些软件反而会比较正确一点（对系统本身的配置而言则相反），所以，若是 X Window 的重度用户，只须知道平时所修改的配置可能会储存到哪一些目录中即可，起码当文件有问题时，可以通过别的主机将所需的数据备份回来。

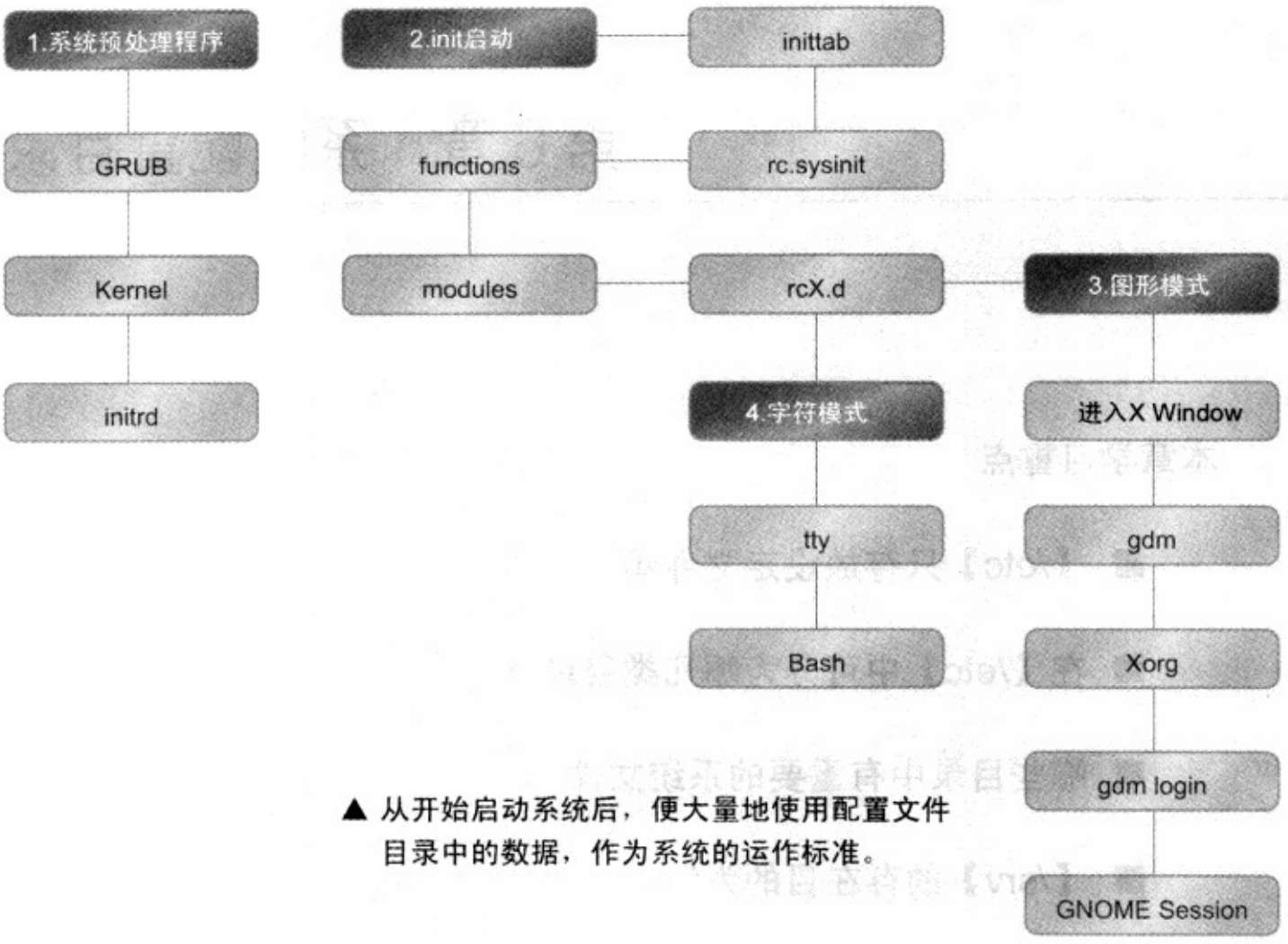
第6章　系统配置目录

本章学习重点

- 【/etc】只存放设定文件吗
- 在【/etc】中可分为哪几类目录
- 哪些目录中有重要的系统文件
- 【/srv】的存在目的为何

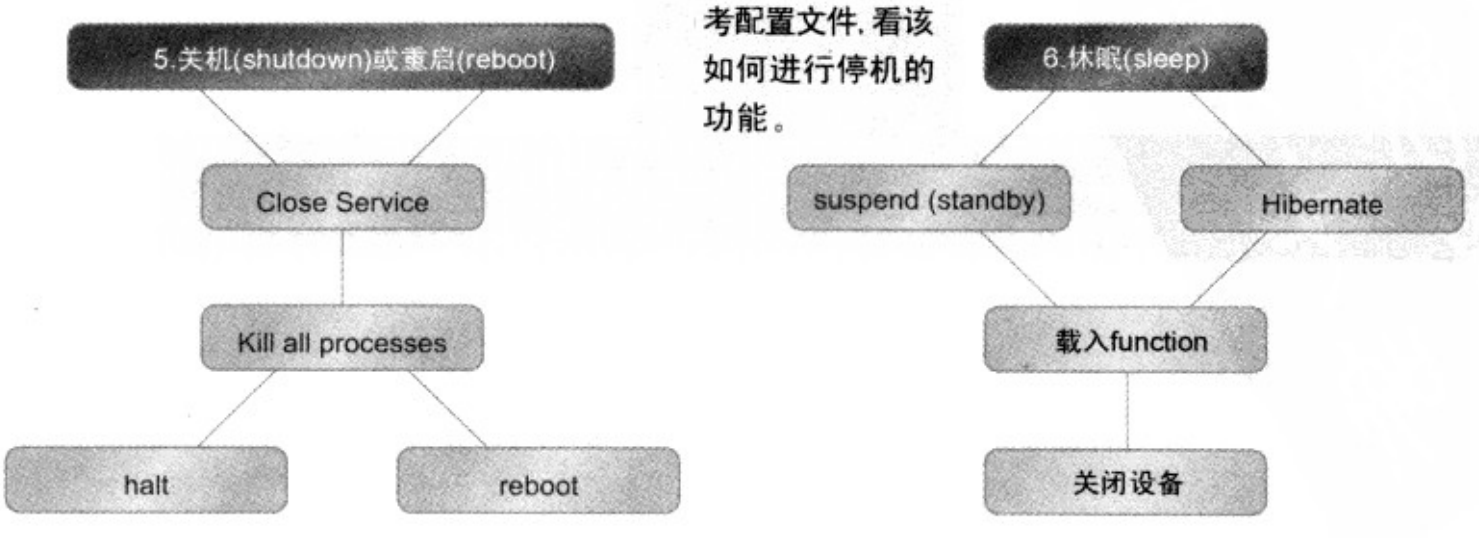
系统流程与章节内容对照图

- 启动流程



▲ 从开始启动系统后，便大量地使用配置文件目录中的数据，作为系统的运作标准。

- 关机流程



▼ 系统要进入关机或休眠时，会参考配置文件，看该如何进行停机的功能。

在一个操作系统中，很多操作都必须通过系统的“配置”文件，以及各项软件的项目配置，才可以正确完成整个系统的启动过程，以及各软硬件的运行，但不论是系统或是各软件的启动结果以及细节，系统必须提供“记录”机制让用户查询，以及供管理员检查各种可能影响系统的因素，这就是本章的重点所在。

本章是全书中很重要的一部分（但笔者反而比较喜欢“第 3 章：Kernel Space 与 User Space 的桥梁——虚拟文件系统”），主要是因为系统大部分会使用到的配置文件，都会在这一章中做介绍（大部分以【/etc】为主），所以一般使用 Linux 的用户，都会将很多学习的时间花在这一章的目录上，但确实这也是非常重要的一件事。

6.1 /etc

【/etc】目录在 Linux 中，是一个非常著名且重要的目录，因为所有用户听过的系统配置文件或服务配置文件，绝大部分都在这一个目录下，但这目录中文件数量的庞大就可想而知（如图 6-1 所示），在笔者系统中/etc 目录中存有近 3000 个目录或文件，这只是一个基本值，因为安装越多的软件，使用越多的功能，数目就会越往上升，所以要完全了解这目录是一大挑战。

```
[root@LinuxTree var]# find /etc/ | wc -l
2935
[root@LinuxTree var]#
```

图 6-1：/etc 目录下的文件数量

基本上，这目录下并没有任何规定要存放哪一类的信息，以下章节是因为/etc 下太多的目录，如果照目录排章节，光是第一层的目录就有 100 多个，所以笔者才会按“配置文件的目的”来分类，让读者可以比较方便找到所需目录或文件名称。笔者简单地将整个/etc 分为以下几类。

1. 基本文件：所有直接放在【/etc】目录下的文件归类为基本文件，但因为文件数目实在太庞大（150 个左右），所以会挑选出经常使用到或是和系统相关度较高的列出，大部分的服务都会列在服务器目录中。
2. 服务器目录：像 samba、http 等服务配置相关目录。
3. 系统目录：像 sysconfig、xen 或网络配置等与系统运行相关的目录。
4. 安全性目录：像 selinux 或 pam.d 等管理系统安全性的目录。
5. X Window 目录：像 X11 或 gdm 管理 X Window 启动或使用上的配置目录。
6. 其他目录：针对单一特殊软件的配置或未能按以上分类方式则放在此目录中，不过，因为软件和各大 Linux 版本的相关性最低，所以有比较多可能只会存在于笔者的 Fedora 系

列和读者的差异会更容易出现。

但因为每一套操作系统都有各自的设计规划，所以，整章中的目录不一定会在每一个操作系统中出现，一定有多有少，甚至有些在笔者示例系统（Fedora7）中的目录，有可能在别的系统中为文件，任何的变化都是有可能的，这要请读者在阅读时注意一下手头的 Linux 版本与文件或目录的变化。

6.1.1 基本文件

在/etc 目录下大部分的独立文件，都是和系统配置直接相关的，所以独立为系统管理部分。

系统管理

◆ aliases

本文件大部分都应用在网络管理员的使用上，因为储存在本文件中的数据（如图 6-2 所示），大都指定哪一些用户的 mail，要转发给 root 管理员，当然在这其中的“原收件人”（左边的名单）与“转发对象”（右边的名单），都是可以自行定义的，这也是该文件最主要的目的。

```
[root@LinuxTree etc]# grep ^[^#] aliases | head -5
mailer-daemon:  postmaster
postmaster:     root
bin:            root
daemon:         root
adm:            root
[root@LinuxTree etc]#
```

图 6-2：aliases 文件中的部分记录

另外一个重点是，列在这个名单中的“原收件人”，不一定要真实的用户，即管理员可以在“不用新增用户的情况下建立一个 mail 账号”，如图 6-3 所示，原本“juergen”用户是寄信给“testuser”，但系统中并没有“testuser”的账号，因此，笔者在【aliases】文件中注记：“将 testuser 转发给 root”，所以最后收到信的人是“root”。

◆ auto.*

【auto.*】所代表的是一系列 autofs 服务所需要的配置文件，这个服务主要是让管理员可以事先定义出一些网络、本机或光驱等默认的路径，使用户可以在不知道任何硬件的路径下，直接使用这些目录，甚至连“mount”命令都无须使用。

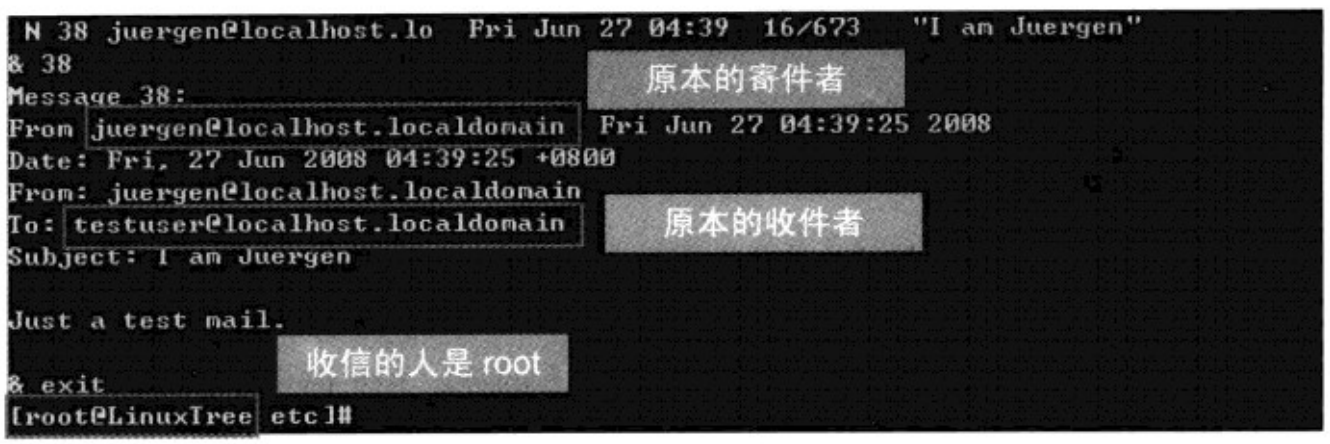


图 6-3：通过 testuser 转发给 root 的示例

这样说或许很抽象，但实际上确实如此，autofs 所需要的文件目前在【/etc】下主要有四个，分别以实际的例子介绍如下，让读者可以明显感受这个服务所带来的便利性。

◆ auto.master

主要的配置文件，负责规划目录的分配与使用（如图 6-4 所示），配置方式为【挂载目录配置文件位置】。目前默认可以提供三种自动挂载模式，分别为以下三种，文件的细节介绍请往下看。

- 磁盘挂载：像一般常使用到的磁盘驱动器、U 盘、光驱等都在此类，默认所使用的目录为【/misc】，这也是系统唯一默认有提供的目录，下面两条配置都是笔者自行加入的。
- 网络磁盘挂载：也就是网络磁盘分享出来的目录，一般最易看到的就是 NFS 的网络磁盘。
- 网上邻居格式挂载：其实就是 Linux 下的 SAMBA 服务，或是其他可支持 CIFS 文件系统的网络共享磁盘。

图 6-4：auto.master 文件的内容

◆ auto.misc

文件中的配置都以实体连接本机的磁盘驱动器为主（如图 6-5 所示），其实文件中的内容有很多是默认的挂载配置，只是被笔者用命令所忽略掉（不然太占篇幅），这里以光驱为例子。

```
[root@LinuxTree etc]# grep ^[^#] auto.misc
cd              -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom
usb             -fstype=ext2                    :/dev/sdb1
[root@LinuxTree etc]#
```

光驱的挂载设定

图 6-5：auto.misc 文件的内容

在图 6-6 中是一个将光驱配置为自动挂载（auto mount）的实际示例，主要有以下几个步骤。

Step 1 一开始系统中只有一个系统默认所建立的目录【/misc】，但里面是空白的，并没有任何文件。

Step 2 此时，如果将 autofs 的服务启动（相关配置当然是要先做好的），就会发现虽然【/misc】目录中没有【cd】子目录，但却可以直接查看其内容列表，其内容已经直接指到光驱的内容。

Step 3 接着就会发现系统已经将【/misc/cd】目录建立好。

Step 4 若 autofs 服务停用，【/misc】中相关的子目录会自动消失，就像一开始的空目录一样。

```
[root@LinuxTree etc]# ls /misc/
[root@LinuxTree etc]# ls /misc/cd
ls: cannot access /misc/cd: No such file or directory
[root@LinuxTree etc]# /etc/rc.d/init.d/autofs start
Starting automount:    OK   ]
[root@LinuxTree etc]# ls /misc/
[root@LinuxTree etc]# ls /misc/cd
                                RPM-GPG-KEY
fedora.css                      RPM-GPG-KEY-beta
GPL                             RPM-GPG-KEY-fedora
                                RPM-GPG-KEY-fedora-rawhide
                                RPM-GPG-KEY-fedora-test
README-BURNING-ISOS-en_US.txt   RPM-GPG-KEY-rawhide
RELEASE-NOTES-en_US.html
                                TRANS.TBL
[root@LinuxTree etc]# ls /misc/
cd
[root@LinuxTree etc]#
```

autofs 服务未启动时是无法使用的

autofs 服务启动后，就可以看到/misc/cd 目录中的内容

/misc/cd 这时才被建立

图 6-6：auto mount 光驱的示例

◆ auto.net

【auto.net】文件和刚刚的【auto.master】及【auto.misc】不太一样，并不是一个配置文件，而是一个 script file，在使用上其实不须做任何调整，只要知道如何使用即可（如图 6-7 所示）。另外，必须注意以下几点。

- 【auto.master】的配置要先确定正确。

- 确认对方 IP 地址，与其 NFS 正确分享出来。
- 要先进入【/net/“IP Address”】（其实也可以用主机名，但这是 auto.master 的配置问题）一次，目录才会被自动产生出来。

```
[root@LinuxTree ~]# showmount -e 10.32.15.28
Export list for 10.32.15.28:
/nfsshare *
[root@LinuxTree ~]# ls /net
[root@LinuxTree ~]# ls /net/10.32.15.28

[root@LinuxTree ~]# ls /net/10.32.15.28/nfsshare/
testfile
[root@LinuxTree ~]#
```

该 IP 所分享出的网络磁盘

通过 autofs 所看到的 NFS 文件系统内容

图 6-7：将 NFS 文件系统自动挂载示例

◆ auto.smb

【auto.smb】文件和刚刚的【auto.net】一样，都是一个 script file，所以使用上没有太大差别，所须注意的地方也都一样，照【auto.net】的方式操作即可，只是将 NFS 转变为 SAMBA 模式（如图 6-8 所示），这里就不再赘述。

```
[root@LinuxTree ~]# smbclient -L 10.32.15.28
Password:
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.0.25-2.fc7]

        Sharename       Type      Comment
        ---------       ----      -------
        samba           Disk
        IPC$            IPC       IPC Service (Samba Server Version 3.0.25-2.fc7
)
Domain=[MYGROUP] OS=[Unix] Server=[Samba 3.0.25-2.fc7]

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
[root@LinuxTree ~]# ls /netsmb/
[root@LinuxTree ~]# ls /netsmb/10.32.15.28

[root@LinuxTree ~]# ls /netsmb/10.32.15.28/samba/
testfile
[root@LinuxTree ~]#
```

该 IP 所分享出的 SAMBA 网络磁盘

通过 autofs 所看到的 SAMBA 文件系统内容

图 6-8：将 SAMBA 或网上邻居的共享磁盘自动挂载示例

◆ bashrc

这是有关用户登录功能配置（若是与系统环境配置或是一些启始化的软件相关的执行配置，则是在【/etc/profile】文件中），也是所有用户一开始所须遵循的文件，如果需要量身订做，可能

必须到主目录中，使用【~/.bashrc】文件，这部分可参考本书“第 5.1.1 节：.bashrc 及.bash_profile”的介绍。

在登录功能配置的【/etc/bashrc】文件中，主要分为以下几个部分（见表 6-1）。

表 6-1：【/etc/bashrc】文件的配置对象与目的

配置对象	配置目的
umask	让系统用户新增或建立文件时，有默认的权限可遵循，默认一般用户（UID 大于 99）为“002”；而管理员（UID 小于等于 99）则是“022”
$PS1	配置以不同方式登录时的【PROMPT 命令】，至于所谓的【PROMPT 命令】也就是用户在 shell 上操作时的【提示符号】，如【[root@linux-tree ~]#】字样
aliases	很多在系统命令使用时的 aliases（别名）都是在这里产生的，因为这一阶段主要就是将所有用户所要使用到的 aliases 配置完毕，像 ls 会出现的颜色、ll 可以取代【ls -l】，以及将 vim 命令对应到 vi，都是在这里所配置的，但这些其实都是在【/etc/profile.d】目录下的配置文件
function	和刚刚一样，都是利用【/etc/profile.d】目录下的配置文件所做的工作，包括一些功能、参数等也都在其中，这也是 bashrc 的最后一个工作

◆ DIR_COLORS 与 DIR_COLORS.xterm

DIR_COLORS 与 DIR_COLORS.xterm 这两个文件都是在配置【ls】命令的颜色（如图 6-9 与图 6-10 所示，经更改后目录的颜色就从蓝色变为没颜色），唯一的差别只是看用户是使用哪一个软件登录的，若是用一般的 tty 登录，就使用【DIR_COLORS】文件，若是在 X Window 下使用“xterm”软件登录，就会改为【DIR_COLORS.xterm】配置文件，当用户一登录系统，该文件的配置就已经存在使用的环境变量了。

该文件的做法，是将内容转变为用户的环境变量存在的，所以，当用户临时变更这个文件时，是不会有实时的效果，必须重新登录，让系统将新的变量值存入环境变量中。

```
[root@LinuxTree ~]# ls
1                                   mbox
anaconda-ks.cfg                     tg3-3.81c.src.rpm
cat                 install.log
dead.letter         install.log.syslog
                    makekernel      system-auth.bak
[root@LinuxTree ~]# _
```

图 6-9：有目录颜色的列表

```
[root@LinuxTree ~]# ls
1                  Documents           mbox              Templates
anaconda-ks.cfg    Download            Music             tg3-3.81c.src.rpm
cat                install.log         Pictures          Videos
dead.letter        install.log.syslog  Public
Desktop            makekernel          system-auth.bak
[root@LinuxTree ~]# _
```

图 6-10：没有目录颜色的列表

◆ fstab

文件原名为 Filesystems Table，存在的意义是可以让管理员将所有需要存在于该系统内的文件系统与目录等的对应关系，全部记载在这个文件中（如图 6-11 所示），让系统于启动时自动加载，换句话说，如果不小心更改了这个文件，且出现错误，则会直接影响到启动的过程。

```
[root@LinuxTree etc]# cat fstab
/dev/VolGroup00/LogVol00 /                       ext3    defaults        1 1
LABEL=/boot              /boot                   ext3    defaults        1 2
tmpfs                    /dev/shm                tmpfs   defaults        0 0
devpts                   /dev/pts                devpts  gid=5,mode=620  0 0
sysfs                    /sys                    sysfs   defaults        0 0
proc                     /proc                   proc    defaults        0 0
/dev/VolGroup00/LogVol01 swap                    swap    defaults        0 0
[root@LinuxTree etc]# _
```

图 6-11：笔者系统中 fatab 文件的内容

图 6-11 为笔者系统中的【/etc/fstab】文件，这里稍微解释表 6-2 中记录内容的意义，以图 6-11 的 sysfs 为例，其意义为“将 sysfs 以 sysfs 的文件系统格式挂载到【/sys】目录中，并不要备份也不需要检查”。

表 6-2：fstab 文件中字段对照表

欲被挂载的目标	挂载目录	文件系统格式	挂载选项	备份频率	检查频率
Sysfs	/sys	sysfs	defaults	0	0

◆ inittab

启动时系统所需要的第一个配置文件【/etc/inittab】最广为人知，因为系统不论要进入哪一个启动阶段或登录时的操作，都可以靠该文件配置完成，但当然它不只做这两件事情。

先看这个文件的全貌（如图 6-12 所示，笔者系统的默认值）：

Ⓐ 初始化操作。

Ⓑ 启动各阶段的服务软件。

Ⓒ 当遇到临时关机或重启操作时的反应。

Ⓓ 各 console 的登录方式。

Ⓔ 进入 X Window 时系统默认使用的管理程序。

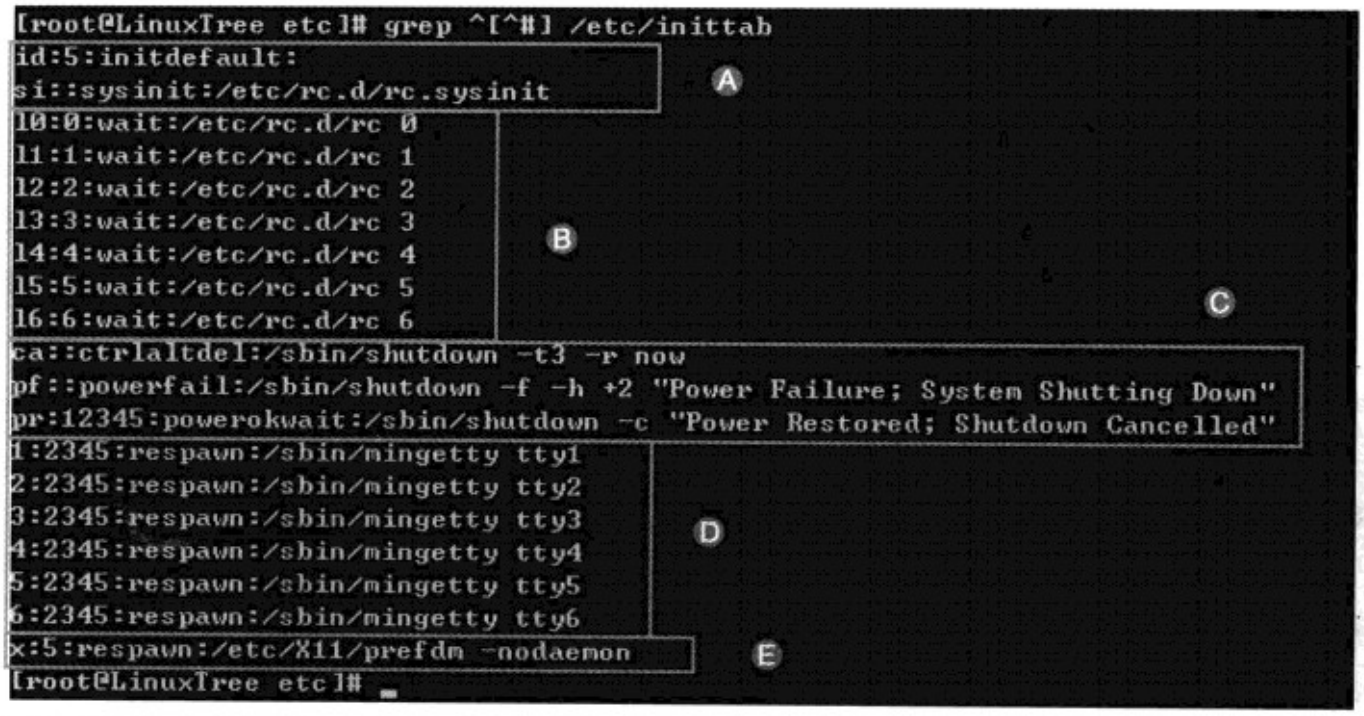


图 6-12：/etc/inittab 文件的内容

其实这个文件并不难懂，重点在于要了解这个文件记录的格式（这里以图 6-12 中 tty1 的登录方式为例）：

id	runlevels	action	process
1	2345	respawn	/sbin/mingetty tty1

该行表示【id】为“1”的记录，只在【runlevel】为“2”、“3”、“4”、“5”时发生，且当执行【process】“/sbin/mingetty tty1”的程序时，只要遇到该程序中断，【action】就“重新启动”（respawn）。

这样大家应该就可以知道【inittab】在做些什么事了。

◆ issue 与 issue.net

用户登录时，都会先看到一个类似欢迎画面，不难发现登录时的信息与【/etc/issue】文件中的内容一模一样（文件中的“\r”与“\m”是从 mingetty 得到的参数，分别代表系统版本与硬件架构），所以 issue 与 issue.net 文件，其实就是登录画面的配置文件，只是 issue 与 issue.net 的使用时机不同，如图 6-13 所示。

- issue：本机登录时使用。
- issue.net：网络登录时使用。

```
Fedora release 7 (Moonshine)
Kernel 2.6.21-1.3194.fc7 on an i686

LinuxTree login: root
Password:
Last login: Fri Jun 27 12:21:56 on tty2
[root@LinuxTree ~]# cat /etc/issue
Fedora release 7 (Moonshine)
Kernel \r on an \m

[root@LinuxTree ~]#
```

图 6-13：登录画面与 issue 文件的比较图

◆ ld.so.conf

该文件就请读者直接参考本章的【系统目录】部分有关【ld.so.conf.d】目录的相关介绍。

◆ localtime

localtime 指的就是“时区”，也就是说，该文件代表着目前系统所使用的时区。在系统中原本就存有各种不同时区的文件，但在【/etc】下只有一个，即当初安装操作系统时所选择的时区（笔者系统为亚洲台北）。

从图 6-14 中可以看到，每一个时区文件的大小是不一样的（图 6-14 显示 Singapore 和 Taipei 的不同），但【/etc/localtime】文件与【/usr/share/zoneinfo/Asia/Taipei】文件则完全相同，因为在【/etc/sysconfig/clock】文件中（原始的配置）已经记录原本所使用的时区就是【亚洲台北】。

```
[root@LinuxTree etc]# grep ZONE sysconfig/clock
# The ZONE parameter is only evaluated by system-config-date.
ZONE="Asia/Taipei"
[root@LinuxTree etc]# ls -l /usr/share/zoneinfo/Asia/Taipei
-rw-r--r-- 2 root root 724 2007-04-03 23:21 /usr/share/zoneinfo/Asia/Taipei
[root@LinuxTree etc]# ls -l localtime
-rw-r--r-- 1 root root 724 2008-02-16 11:44 localtime
[root@LinuxTree etc]# ls -l /usr/share/zoneinfo/Asia/Singapore
-rw-r--r-- 2 root root 402 2007-04-03 23:21 /usr/share/zoneinfo/Asia/Singapore
[root@LinuxTree etc]#
```

图 6-14：localtime 文件的源文件

所以，若时区要换掉，只要将原本的【localtime】文件换为时区目录中（/usr/share/zoneinfo）的其他时区文件（像 Singapore），再把【/etc/sysconfig/clock】文件的配置为该时区，这样重启后就生效了（如图 6-15 所示，时间其实早在时区文件替换时就已经不一样了）。

图 6-15：时区的变换

◆ motd

这文件的全名是“message of the day”，主要的用意是让所有“成功登录”的用户都可以看到该信息（如图 6-16 所示）。

这点很重要，因为如果不考虑登录的成功与否，“issue”或“issue.net”就可以办到，所以 motd 的主要目的，就是在【登录后到执行 shell 的这一段时间】将信息通知所有用户。

图 6-16：motd 文件的作用

◆ mtab

还记得 fstab 文件吗？系统在启动时会自动将该文件中所有挂载的文件系统一并完成，其实在挂载完成时会有一个小操作，就是将成功挂载的文件系统，写入【/etc/mtab】文件中（如图 6-17 所示）作为一个记录，所以这个文件可以当作检查挂载状况的记录文件，其实和【mount】命令所呈现的结果也应该是一样的，因为都是目前所挂载的文件系统。

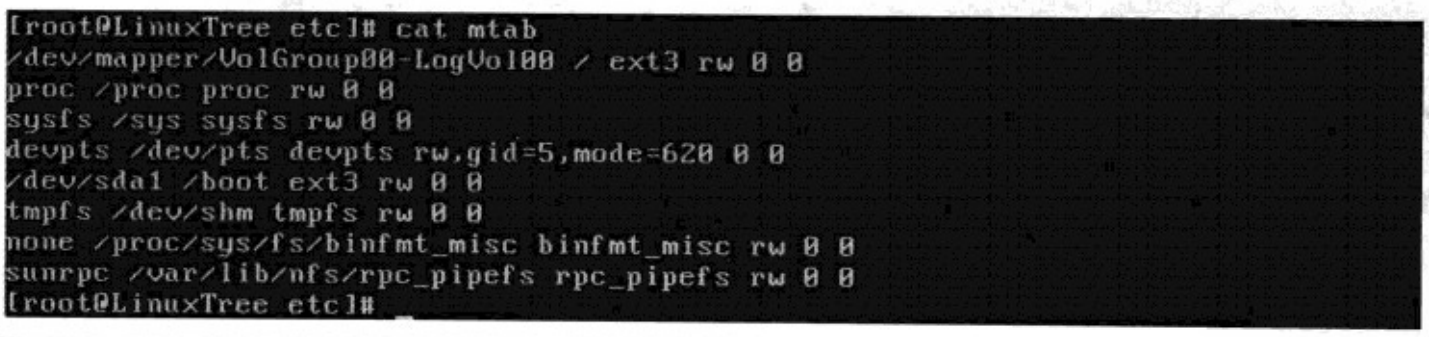


图 6-17：mtab 文件的内容

◆ prelink.conf

Linux 系统中有许多共享函数库，当程序执行时，会动态地和这些文件系统中的函数库文件

做连接，这个操作就叫做【link】。而这一个文件的目的在于，当程序需要大量的连接时，势必会拖慢整体的进度（少的时候可能没多大感觉），为了提升共享函数库的使用效率，因此就有了一个【prelnk】的程序，协助软件执行前，就预先做好事前的【link】操作，当然，这个文件就是记录有哪一些执行文件和函数库是需要预先连接的。

◆ profile

【/etc/profile】文件是当用户登录时，使用 bash 所读入的第一个配置文件，接下来才是之前提到的【/etc/bashrc】，这个优先级是很重要的。虽然两者都是 script file，但在这两个文件存在的定义上是不同的。

- 【profile】：系统相关的环境配置或一些初始化的软件执行配置。
- 【bashrc】：偏向功能或别名的配置则记录在 bashrc 中。

因此，【profile】文件中主要的功能见表 6-3。

表 6-3：【profile】文件中主要的功能

配置对象	配置目的
$PATH	默认路径的配置，以及区分普通用户与管理员默认路径的差异
$HOSTNAME	配置主机名
$HISTSIZE	配置在 shell 下所要记录的历史命令条数，默认大小都以 1000 条为基本值
键盘的对应	有点像是 DOSKEY，这里配置的部分就是像用户会用到的【Tab】键功能，很多都可以在这一个阶段做调整，包括有时很讨厌的小喇叭声音
function	和 bashrc 一样，都是利用【/etc/profile.d】目录下的配置文件所做的工作，包括一些功能、参数等都在其中，但因为 bashrc 中也有，所以还是要以最后 bashrc 所执行的结果为准

◆ screenrc

这一个文件其实是一个【screen】软件的配置文件，原本不想放进来，但笔者觉得这功能实在是太棒了！以前没用过不知道，用过才叫好。正常在一个 console 下（像 tty1）就是利用“这一个”窗口处理事情，但【screen】软件可以帮用户将一个窗口变为多个窗口（如图 6-18 所示，在这里先记得在 shell 下已经通过 screnn 软件开了九个 ssh 的软件，下面会再提到），不论是在 X Window 下的 tab 或是 shell 下的 console，都没有问题。

```
[root@LinuxTree ~]# pstree|grep screen
      |-gnome-screensav
      |-gnome-terminal-+-bash---screen---screen---ssh
      |-login---bash---screen---screen---9*[ssh]
[root@LinuxTree ~]#
```

图 6-18：screen 在 X Window 与 shell 下产生的程序

X Window 的部分就不多说，因为笔者觉得在 shell 下才厉害，别人会吓一跳（可能最后只是笔者自己吓一跳吧，因为没用过），但用法基本上都是一样的，只是显示上有些小的差异。

配置非常简单，笔者只在【/etc/screenrc】文件的最下方加入如图 6-19 中的配置，接着再执行【screen】命令，就可以进入【screen】的画面。

```
[root@LinuxTree ~]# tail -10 /etc/screenrc
screen -t demo 0 ssh localhost
screen -t test 1 ssh localhost
screen -t test 2 ssh localhost
screen -t test 3 ssh localhost
screen -t test 4 ssh localhost
screen -t test 5 ssh localhost
screen -t test 6 ssh localhost
screen -t test 7 ssh localhost
screen -t test 8 ssh localhost

[root@LinuxTree ~]#
```

笔者设定的 9 个 screen，框起来的部分是窗口的名称和 id

图 6-19：笔者在 screenrc 文件中加入的内容

一开始只会看到唯一的一个【ssh】命令画面（但是这是配置中的最后一个，也就是 id 为 8 的 ssh），但不要担心，只要再按【Ctrl+A】，紧接着再按【Shift+ "】，就会出现这个菜单了（看起来很怪，但实际的快捷键就是如此），看到这菜单，就应该很清楚【screen】命令的目的了（如图 6-20 所示），这也是【screen】命令让笔者觉得很好玩的地方，供读者参考。

```
Num Name                                                    Flags
  0 demo                                                    $
  1 test                                                    $
  2 test                                                    $
  3 test                                                    $
  4 test                                                    $
  5 test                                                    $
  6 test                                                    $
  7 test                                                    $
  8 test                                                    $
```

很像在使用 KVM 所出现的菜单画面

图 6-20：screen 执行的菜单画面

◆ securetty

securetty 文件中全部都是登录接口的名字（像 tty1、tty2，等等），因此就不在这里列出，【/etc/securetty】文件主要是 login 程序在使用的，只要是列在该文件中的接口，就表示是可以使用的接口；当然，反过来说，若从列表中被移除，就会无法使用该接口。

如图 6-21 中的登录失败画面，并不是因为密码打错，而是笔者将 tty2 的记录从【/etc/securetty】文件中移除，所以在 tty2 登录时，由于不属于 login 所允许的 console 而被拒绝登录。

```
Fedora release 7 (Moonshine)
Kernel 2.6.21-1.3194.fc7 on an i686

LinuxTree login: root
Password:
Login incorrect
```

图 6-21：被 securetty 限制住无法登录的画面

◆ shells

这是一个比较单纯的文件，里面只是记载目前系统所拥有 shell 种类的路径，但该文件不是被系统所直接读取的，而是通过一个【chsh】更换 login shell 环境的命令所使用的，当用户使用【chsh】命令切换默认的 login shell 时，该命令就会将新的 shell 和此文件做对比（包括完整路径），如图 6-22 所示。

```
[root@LinuxTree etc]# cat shells
/bin/sh
/bin/bash
/sbin/nologin
/bin/tcsh
/bin/csh
/bin/zsh
[root@LinuxTree etc]#
```

图 6-22：shells 文件的内容

◆ sudoers

一般用户在 Linux 系统环境下，若希望拥有管理员的权限，除了将身分切换为“root”之外，还有一个做法，就是使用【sudo】命令，直接取得“root”的权限使用文件。

但不是每一个用户都可以使用【sudo】命令，因为只要是列在其中的，可以不需要知道 root 的密码，直接操作 Linux，所以【/etc/sudoers】文件就是在配置权限的分配方式，以免过于混乱。

参考图 6-23，若【juergen】是列在【sudoers】文件中且权限被打开，自然就可以使用【sudo】命令，但如果把【sudoers】文件中的【juergen】记录移除，就会立即出现错误。

```
[root@LinuxTree etc]# grep juergen sudoers
juergen ALL=ALL
[root@LinuxTree etc]# su - juergen
[juergen@LinuxTree ~]$ sudo ls /etc/profile
/etc/profile
[juergen@LinuxTree ~]$ sudo ls /etc/profile
juergen is not in the sudoers file.  This incident will be reported.
[juergen@LinuxTree ~]$ _
```

juergen 在 sudoers 文件中的设定被移除了

图 6-23：sudoers 文件对用户的影响

◆ sysctl.conf

【/etc/sysctl.conf】文件主要是帮用户配置【/proc/sys】目录下所有文件的值（通过【sysctl】命令，有关【/proc/sys】的目录请参考“第 3.2.10 节：/proc/sys”），方便用户以最常用的配置文件配置方式，去改变【/proc/sys】对系统产生变化，最大的好处应该还在于“只需要做一次的配置”，不然，直接使用【echo “1” > /proc/sys/net/xxx】的命令就可以取代【sysctl】的功能了。

最常看到的【sysctl.conf】应用（如图 6-24 所示），就是当用户在使用 NAT 时，都需要将【/proc/sys/net/ipv4/ip_forward】文件的值设为“0”，让 IP 数据包可以通过本机的一张网卡到另一张，这是目前笔者最常看到大家使用的一项功能，但其实里面有非常多的系统相关文件可以做配置。

在图 6-24 中，可以看到所有【sysctl.conf】文件中的配置，都是指到【/proc/sys】目录下的文件，用户可以自行新增或删除里面的任何记录，而且有一点要注意，就是其系统默认为“不会”执行【sysctl】这个命令，需要用户自行执行（配置完成后可使用【sysctl -p】直接让系统使用【sysctl.conf】文件中的配置）。

```
[root@LinuxTree etc]# grep ^[^#] sysctl.conf
net.ipv4.ip_forward = 0
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
kernel.sysrq = 0
kernel.core_uses_pid = 1
net.ipv4.tcp_syncookies = 1
[root@LinuxTree etc]# ls /proc/sys/

[root@LinuxTree etc]# cat /proc/sys/net/ipv4/ip_forward
0
[root@LinuxTree etc]#
```

图 6-24：sysctl.conf 文件对/proc 目录下的改变

◆ syslogd.conf

这个文件其实也是一个 syslogd 服务的配置文件，会加进来是因为对系统有很大的帮助，因为在系统管理中，记录文件非常重要，而系统相关的记录文件主要来自于两大资源——klogd 和 syslogd（这两个文件的主要差异，请读者自行参考笔者的另一著作《Linux 操作系统之奥秘》的第 5.3 节），简单地说，klogd 负责启动时的记录，而 syslogd 负责系统服务的记录。

所以在 syslogd 服务中的配置文件【/etc/syslog.conf】（如图 6-25 所示），如果配置得好，是可以看到更多的信息，有时候系统发生问题，或许不是没有记录，而是因为系统记录的范围比较小，或是根本没有要求系统记录下来。

```
[root@LinuxTree etc]# grep ^[^#] syslog.conf
*.info;mail.none;news.none;authpriv.none;cron.none              /var/log/message
s
authpriv.*                                                      /var/log/secure
mail.*                                                          -/var/log/maillog
cron.*                                                          /var/log/cron
*.emerg                                                         *
uucp,news.crit                                                  /var/log/spooler
local7.*                                                        /var/log/boot.log
news.=crit                                                  /var/log/news/news.crit
news.=err                                                   /var/log/news/news.err
news.notice                                                 /var/log/news/news.notice
[root@LinuxTree etc]#
```

图 6-25：syslogd.conf 文件中的内容

以图 6-25 中的 cron 服务的记录配置为例，每一条记录的规则可以分为三部分——【cron】、【*】和【/var/log/cron】，分别对应到 3 个主要的配置文件字段——【facility】、【priority】与【action】。

若读者有任何不清楚或希望了解更高级一点的内容（配置细节还有非常多没有列出来，如图 6-25 中的“news.=err”只要 err 等级，而一般只有“.”代表此等级之上都要记录，这里仅以基本的字段解释），可直接参考【man 3 syslog】中的叙述（不过在 syslog 的描述中，字段名称有些许的差异，但不影响其正确性），笔者将其整理过以中文方便阅读，如表 6-4 所示。

表 6-4：字段解释

字段名称	目的	可支持模式
facility	syslogd 所支持的服务名称	1. auth：安全性或需要用到用户认证上的问题，但以后的版本可能会以下面的 authpriv 为主 2. authpriv：和 auth 一样，但比较注重系统安全及认证上信息的隐密性 3. cron：主要记录 cron 和 at 两个任务计划软件的信息 4. daemon：普通系统服务的信息 5. ftp：针对 ftp 服务的信息 6. kern：记录 kernel 的信息 7. local0~local7：保留给本机所使用 8. lpr：打印机的信息 9. mail：Mail 系统所发出的信息 10. news：新闻相关服务的信息 11. syslog：系统内部 syslogd 服务本身所产生出来的信息 12. user：由普通用户等级所执行的程序产生出的信息 13. uucp：UUCP 工具所产生出的信息

续表

字段名称	目的	可支持模式
priority	信息记录的等级，也就是详细程度，由下往上是代表问题越来越严重。	1. emerg：系统已经处于不稳定的状态 2. panic：和 emereg 是一样的 3. alert：该条记录需要立即取得，代表情况是非常严重的 4. crit：critical issue，就是发生重要的问题 5. err：错误状况发生 6. error：和 err 是一样的 7. warn：警告信息，但通常警告不一定代表发生错误 8. warning：和 warn 是一样的 9. notice：系统正常，但有需要被注意的信息 10. info：普通信息 11. debug：错误检测所需的信息
action	当信息产生时该储存在哪里	1. 记录在普通文件 2. pipe 文件，记录会以 FIFO 的方式记录，完全不会占用文件空间，但需要实时的监控 3. 从终端机或 console 出现 4. 送到远程主机，但这也同时也需要远程主机的支持 5. 送给某些用户，一般都会转发给 root 6. 发送给目前在线登录的所有用户

笔者以一个很单纯的登录记录做示例（如图 6-26 所示），在记录方法上，因为 pipe 文件是最新的方式，故采用 pipe 文件的方式记录。

- 读者先自行通过【mkfifo /var/log/secure.pipe】命令建立该 fifo 文件。
- 在【/etc/syslog.conf】文件中先做配置（当然 syslogd 服务要重新启动或加载）。
- 使用“cat”命令监控该文件，或许有读者会觉得用“tail -f”比较方便，但因为 fifo 文件的特性就是完全不会存在文件中，所以“tail -f”在该文件中是看不到任何信息的。

```
[root@LinuxTree etc]# grep pipe syslog.conf
authpriv.*                                              i/var/log/secure.pipe
[root@LinuxTree etc]# cat /var/log/secure.pipe
Jun 28 06:54:35 LinuxTree login: pam_unix(login:session): session opened for use
r juergen by LOGIN(uid=0)
Jun 28 06:54:35 LinuxTree login: LOGIN ON tty4 BY juergen

[root@LinuxTree etc]# ls -l /var/log/secure.pipe
prw-r--r-- 1 root root 0 2008-06-28 06:54 /var/log/secure.pipe
[root@LinuxTree etc]#
```

这里是 juergen 用户在另一个 tty 接口登录的信息

图 6-26：利用 syslog.conf 文件配置记录方式

- 可看到信息后，该记录文件的大小还是“0”。

网络管理

这里的网络管理和一般网站是没有关系的，主要是包含一些主机网络的相关配置，不会和其他软件有太大的关系。

◆ host.conf

主机名解析的配置值（如图 6-27 所示），里面注明系统在接受到主机名时，解析的方式及顺序是什么（图中由左至右，所以是查询 hosts，再通过 bind 查询），一般默认通过【/etc/hosts】文件（图 6-27 的 hosts）和 DNS（图 6-27 的 bind），目前可支持的除这两个之外，还有一个就是 NIS。

```
[root@LinuxTree etc]# cat host.conf
order hosts,bind
[root@LinuxTree etc]# _
```

图 6-27：host.conf 文件的内容

◆ hosts

Linux 系统会将所有需要的主机名储存在 hosts 文件中，接续上一个文件（host.conf）中的内容，其中所配置的“hosts”，指的就是【/etc/hosts】文件，所以查询该文件的优先级高过 DNS，也就是说，出现在【/etc/hosts】文件中的主机名，一开始就会被系统接受，若有和 DNS 重复的，一样以这个文件为主，因为有记录在其中的就根本不会问 DNS。

由图 6-28 中可以知道，此主机的主机名为 LinuxTree、localhost.localdomain 或 localhost，也就是说，即使 DNS 没有 LinuxTree 这台主机，但只要文件中有这个名称，系统就自动接受，且 IP 为“127.0.0.1”，当然管理员可以自行增删所有的主机及 IP 的对应记录。

```
[root@LinuxTree etc]# cat hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       LinuxTree       localhost.localdomain   localhost
::1     localhost6.localdomain6 localhost6
[root@LinuxTree etc]#
```

图 6-28：hosts 文件内容

◆ hosts.allow 与 hosts.deny

这两个文件牵涉到一个 Linux 网络安全性的机制【TCP Wrapper】，概念很像我们之前提过的 at.deny 或是 cron.deny 机制，主要以【hosts.allow】与【hosts.deny】控制网域、网段或计算机可使用的本机服务。

【TCP Wrapper】的主要流程如图 6-29 所示。

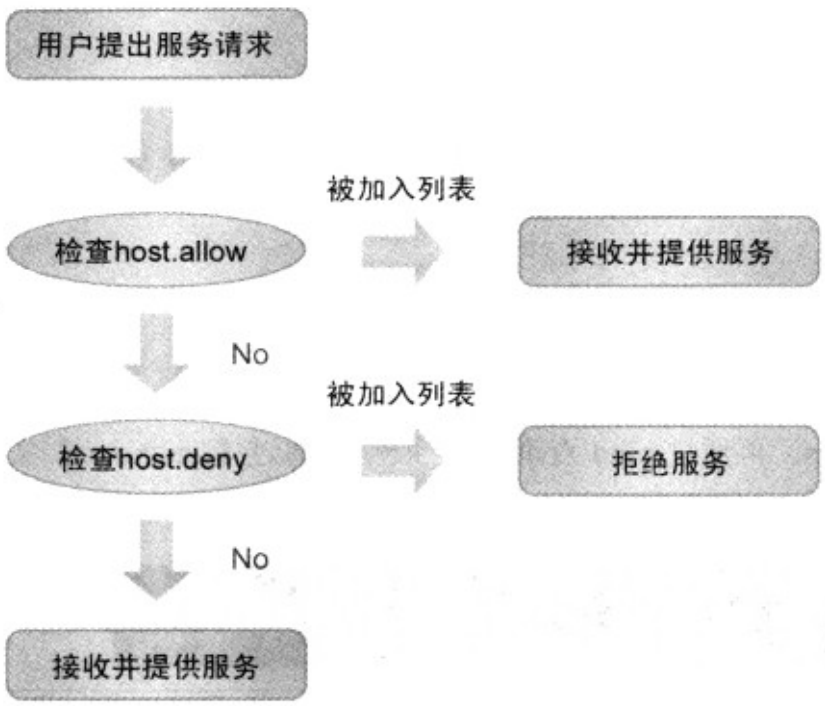


图 6-29：TCP Wrapper 的流程图

从图 6-29 中可以看到，这两个文件主要负责帮系统过滤掉使用服务的主机或用户，进而保障系统的安全，但要注意的事，并非所有的服务都可以使用，目前可使用 TCP Wrapper 机制的服务，较常见的像 telnet、pop3、vsftpd、sshd 等，特别是 xinetd 所负责管理的服务，就应该可以通过 TCP Wrapper 控制。

笔者用一个 sshd 做一个简单的例子（如图 6-30 所示）。

图 6-30：用 hosts.allow 与 hosts.deny 搭配出的安全性

- 当【hosts.allow】允许本机，且【hosts.deny】拒绝所有人时，本机“可以”使用 ssh 联机到本机服务。
- 当【hosts.allow】不设规则，且【hosts.deny】拒绝所有人时，本机“无法”使用 ssh 联机到本机服务。

◆ nsswitch.conf

该文件主要记录系统应如何查询主机名、密码、用户组、网络等，或是查询顺序的编排。如图 6-31 中文件的内容，常会注意的就是选出的三条记录。

- passwd：查询方式为直接以系统中的文件为主（/etc/passwd）。
- hosts：查询方式为先通过系统中的文件（/etc/hosts），再查询 DNS 系统（端视系统的配置，像 Hinet 的 168.95.1.1）。
- services：服务的编号应查询系统的文件（/etc/services）。

```
[root@LinuxTree etc]# grep ^[^#] nsswitch.conf
passwd:     files
shadow:     files
group:      files
hosts:      files dns
bootparams: nisplus [NOTFOUND=return] files
ethers:     files
netmasks:   files
networks:   files
protocols:  files
rpc:        files
services:   files
netgroup:   nisplus
publickey:  nisplus
automount:  files nisplus
aliases:    files nisplus
[root@LinuxTree etc]#
```

图 6-31：nsswitch.conf 文件的内容

◆ resolv.conf

这其实就是 Linux 系统内 Client 端的 DNS 及域名配置文件，不论是固定 IP 或是 DHCP 的方式，都会记录在这个【/etc/resolv.conf】文件中，图 6-32 中，因为笔者所使用的是 DHCP，所以 DHCP 的 Client 端软件，会自动在文件中加注，以表明是该软件所记录的。

```
[root@LinuxTree etc]# cat resolv.conf
; generated by /sbin/dhclient-script
search notes.foxconn.com
nameserver 10.39.7.32
nameserver 10.36.3.30
nameserver 10.36.3.19
[root@LinuxTree etc]#
```

图 6-32：resolve.conf 文件内容

◆ services

每个网络程序执行的时候，都会有一个默认的 port number（端口号），像浏览器软件（这台系统默认是 Firefox）默认就会连到对方计算机的“80”port，但这个“80”是哪来的？在 Linux

就是由【/etc/services】文件所提供的。

像图 6-33 中的 services 文件内容，原本 ssh 的 port 编号为“22”，但笔者将其改为“2222”，结果在使用 ssh 联机时，系统自动以“2222”port 连到远程计算机，自然就会失败。

```
[root@LinuxTree etc]# grep 2222 services
ssh             2222/tcp                        # SSH Remote Login Proto
col
rockwell-csp2   2222/tcp                        # Rockwell CSP2
rockwell-csp2   2222/udp                        # Rockwell CSP2
[root@LinuxTree etc]# ssh localhost
ssh: connect to host localhost port 2222: Connection refused
[root@LinuxTree etc]#
```

图 6-33：services 文件中所有有关 ssh 的记录

◆ xinetd.conf

服务启动的方式一般分为以下两种。

- standalone 模式：像一般 http 或是其他服务，启动时自己本身会产生父程序，并不需要别的程序协助启动，所以在等待提供服务时，都是由本身的程序来监听网络状态的。
- inetd 模式：有点像是托管模式，在 inetd 下的服务并不会自己启动，唯一一个会在在线监听网络状态的，是 xinetd 服务，这服务若听到某一个子服务（像 telnet）需要启动，必须要通过 xinetd 协助启动，所以 telnet 只是 xinetd 下的一个子程序。

而【xinetd.conf】就是 xinetd 的主配置文件（如图 6-34 所示），其最重要的目的，就是为【xinetd.d】下所有的子服务建立一个标准的规范使其可以遵循，所以是每个子服务会使用的统一配置。目录下的文件，可以引用。

```
[root@LinuxTree etc]# grep ^[^#] xinetd.conf
defaults
{
        log_type        = SYSLOG daemon info
        log_on_failure  = HOST
        log_on_success  = PID HOST DURATION EXIT
        cps             = 50 10
        instances       = 50
        per_source      = 10
        v6only          = no
        groups          = yes
        umask           = 002
}
includedir /etc/xinetd.d

[root@LinuxTree etc]# ls xinetd.d/
chargen-dgram   daytime-stream  echo-stream     telnet
chargen-stream  discard-dgram   ftp             tftp
cvs             discard-stream  rsync           time-dgram
daytime-dgram   echo-dgram      tcpmux-server   time-stream
[root@LinuxTree etc]#
```

这是每个子服务可被联机的数目

目前 xinetd 可支持的子服务

图 6-34：xinetd.conf 的内容与 xinetd.d 目录下的文件

服务管理

只要是普通用户会听到的服务器相关的独立配置文件，都会放在该目录下。

◆ anacrontab

这个文件是属于一种任务计划软件的配置文件，一般管理员常使用的任务计划软件，大都以crond 为主，【anacrontab】软件和 crond 其实有点相辅相成，crond 负责任务计划，而 anacron 则是负责以“间隔多久”为主要的目标，刚好可以补足 crond 所做的周期性安排。其间隔的判断方式是以存放在/var/spool/anacron/目录中的时间戳记（timestamp）为主要的依据，来决定是否要开始执行。

anacron 与 crond 最大的差别在于，若以 crond 为主要的任务计划方式，会有一个小问题，如果我们先假设用户以 crond，配置每周六、日凌晨 12 点到 3 点要进行更新软件的操作，但刚好却在假日或任务计划时间内停电，计算机在未启动的状态下渡过任务计划时间，这些更新的操作就会持续延到下星期，对有些系统管理人员来说会是很大的困扰，尤其像备份之类的工作，因此，若以 anacron 补足这方面的缺点，就会比较完整，而 crond 与 anacron 是可以并行的。

这个配置文件的内容（如图 6-35 所示）和 crontab 的形式很像，主要是配置两个数字：周期天数和延迟时间（以 cron.daily 为例，为 1 天和 65 分钟）。当周期天数到了，anacron 会判断该程序是否已执行过（1 天），若没有，则会等待一段时间再执行，这段时间就是配置好的延迟时间（65 分钟）。

```
[root@localhost X11]# cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

1       65      cron.daily              run-parts /etc/cron.daily
7       70      cron.weekly             run-parts /etc/cron.weekly
30      75      cron.monthly            run-parts /etc/cron.monthly
[root@localhost X11]#
```

图 6-35：anacrontab 配置文件的内容

◆ at.deny

at 也是一个任务计划软件，但主要针对单一时间做配置，也就是用户可以定义某个时间执行某一件事，但这件事只会发生一次。而【at.deny】文件就是该服务的拒绝列表，也就是黑名单，只要被记录在其中的用户，就无法使用 at 所提供的任务计划服务。如图 6-36 所示，原本“juergen”

用户是可以使用 at 功能的，但 root 将“juergen”的名字加入【at.deny】文件后，就无法再进行任何的任务计划编辑，但所有的前提当然就是 at 的服务是要在启动中的。

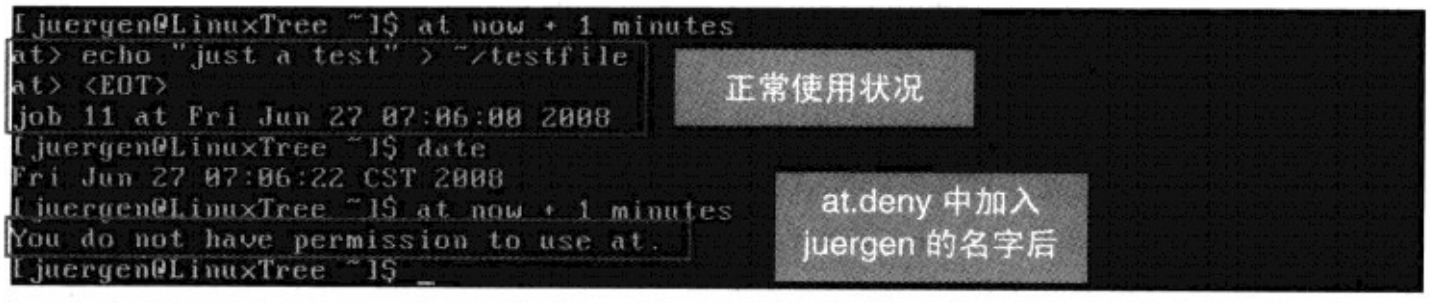


图 6-36：at.deny 对用户的影响

at 服务除了有黑名单外，也有白名单，也就是【at.allow】，两个文件刚好是相反的，不过特别要注意的是“没记录在 at.allow 中的用户，就是被拒绝的对象”（若是空文件代表全部接受）。Redhat 默认只有【at.deny】文件，如果需要【at.allow】文件，则必须自行建立。

◆ crontab 与 cron.deny

这是 cron table 的主配置文件，cron table 有点像是 Windows 的 schedule 软件，可以让用户定义年、月、日、时、分、秒，以决定要在何时进行周期性的软件，配置方式简单明了，也有很多软件默认都会使用 crond 来执行。crond 默认会执行的文件可以参考/etc/crontab 文件中所列的列表（如图 6-37 所示），从图中可以看到其实周期性执行的是在/etc 目录下的 cron.hourly、cron.daily、cron.weekly，以及 cron.monthly 目录中的所有文件，配置方式也很简单，只要按照分、时、日、周、月的顺序配置便能排出用户想要的任务计划。

至于 cron.deny 与之前提过的 at.deny 大同小异，作法都是一样，同样都能自行建立 cron.allow，而 cron.deny 是默认系统所建立的，读者要了解这个文件就请自行参考本节所介绍的【at.deny】文件。

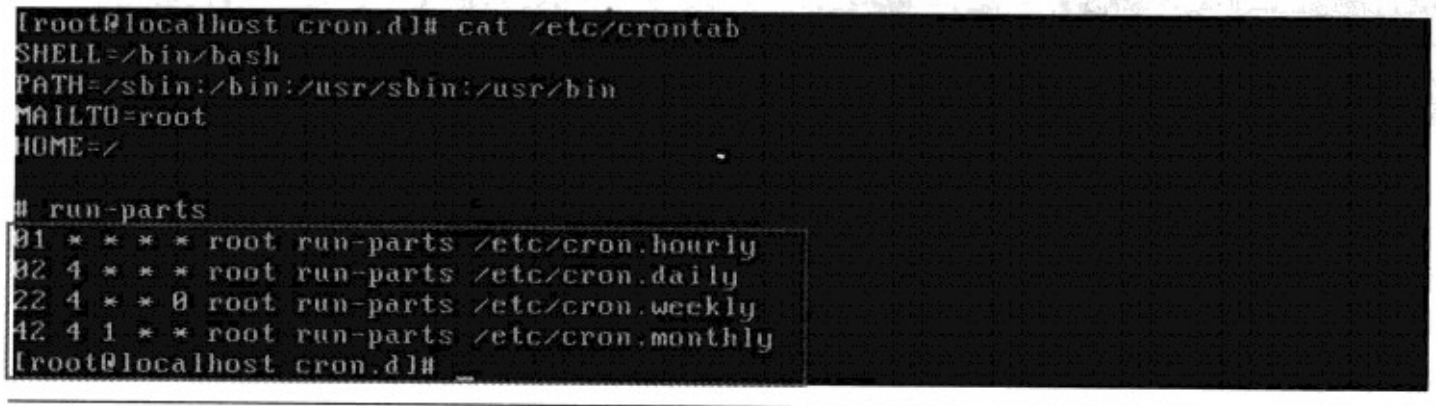


图 6-37：crontab 文件的内容

◆ exports

【exports】文件是 NFS 服务的主要配置文件，主要目的就是将本机的目录共享到网络上，供其他人使用，Client 端可以手动或配合 autofs 的方式，将该目录挂载到 Client 端的文件系统上。其实在配置上也有非常多的选择方式。

但一来笔者懒得做太多的配置（毕竟不是网管人员，没必要搞得太复杂），二来笔者不喜欢记太多的配置细节（要记的话，Linux 永远记不完），三来这里只是要介绍这个文件，所以，仅将一般笔者常使用的配置方式介绍给读者（如图 6-38 所示），方便好用，也不用考虑权限的问题（这有时是大坏处，请斟酌），图中配置的意义为“全部的用户都可以读取和执行本机“/nfsshare”目录中的数据，并同步执行”。

```
[root@LinuxTree etc]# cat exports
/nfsshare       *(rw,sync)
[root@LinuxTree etc]#
```

图 6-38：exports 文件的内容示例

账号管理

在目录中的单一文件，有许多其实都和系统内用户管理有很大的关系，因此，将所有与账号相关的配置文件全部放到此节中，方便用户整合使用。

◆ group 与 gshadow

这两个文件放在一起，是因为它们都属于用户组管理的范围。大部分读者应该都很熟悉【group】文件，里面记载着（如图 6-39 所示）系统内所有用户组的数据，分成下列几个部分：

- 用户组名称。
- 用户组的密码。
- GID 编号。
- 附属在该用户组下的用户名称。

```
[root@LinuxTree etc]# head -5 group
root:x:0:root
bin:x:1:root,bin,daemon
daemon:x:2:root,bin,daemon
sys:x:3:root,bin,adm
adm:x:4:root,adm,daemon
[root@LinuxTree etc]#
```

图 6-39：group 文件的部分内容

在图片中有“x”的记号，这和【passwd】文件的意义相同，代表用户组的密码，但要解释为什么要有用户组的密码，就有点小复杂了。

若现在有两个用户【juergen】与【newuser】，在默认的情况，当然这两个人的用户组分别为【juergen】与【newuser】，因为系统默认在建立用户时，就已经建立了默认的用户组，换句话说，这两个用户所建立的文件，不论是【所有者】或【拥有用户组】，一定都是【juergen】与【newuser】（如图 6-40 所示），这是众所周知的事实。

```
[juergen@LinuxTree ~]$ touch file
[juergen@LinuxTree ~]$ ls -l file
-rw-rw-r-- 1 juergen juergen 0 2008-06-27 14:59 file
[juergen@LinuxTree ~]$
```

图 6-40：juergen 所产生文件的属性

但如果发生一种情况，当【juergen】这个人突然要使用【newuser】的用户组当作【juergen】的默认用户组，就需要利用用户组密码做切换，当然，前提是管理员要先替该用户组配置好密码（可使用“gpasswd newuser”命令，而这个密码就储存在【gshadow】文件中（如图 6-41 所示）），并告知【juergen】所配置的密码。【gshadow】文件分为以下字段。

- 用户组名称。
- 用户组密码，这里才真正存有密码值。
- 用户组管理员账号。
- 附属在该用户组下的用户名称，所以和刚刚 group 文件应该要一致。

```
[root@LinuxTree etc]# tail -5 gshadow
xfs:x::
tclhttpd:x::
gdm:x::
juergen:!::
newuser:$1$3gl1U2AN$gQddJr6/lo9eeItIfeEMG1::
[root@LinuxTree etc]#
```

密码已经过处理

图 6-41：gshadow 文件中的内容

在【juergen】取得密码后，就可以利用【newgrp】命令，将本身的“默认用户组”由【juergen】切换为【newuser】（如图 6-42 所示），这时所产生的任何文件或做的任何操作，其用户组权限都是以【newuser】为主的，当然也可以直接跳回原本【juergen】用户组，只须要离开【exit】即可。

◆ login.defs

这其实和文件名中的 login 比较没多大关系，重点在建立账号时的一些限制，也就是系统在建立账号时所参考的配置。从图 6-43【login.defs】文件的内容，可以看到都是和账号初始配置相

关的，像笔者所框选出的三个配置，分别为：“email 文件的存放目录为‘/var/spool/mail’”、“UID 的最小值为 500”与“账号建立时一并建立目录”。

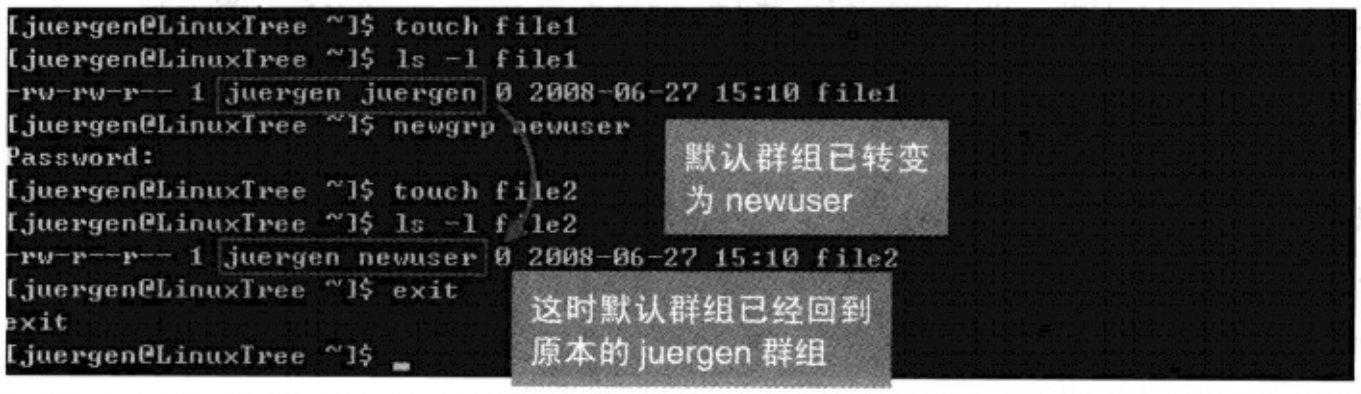


图 6-42：切换用户组的方式

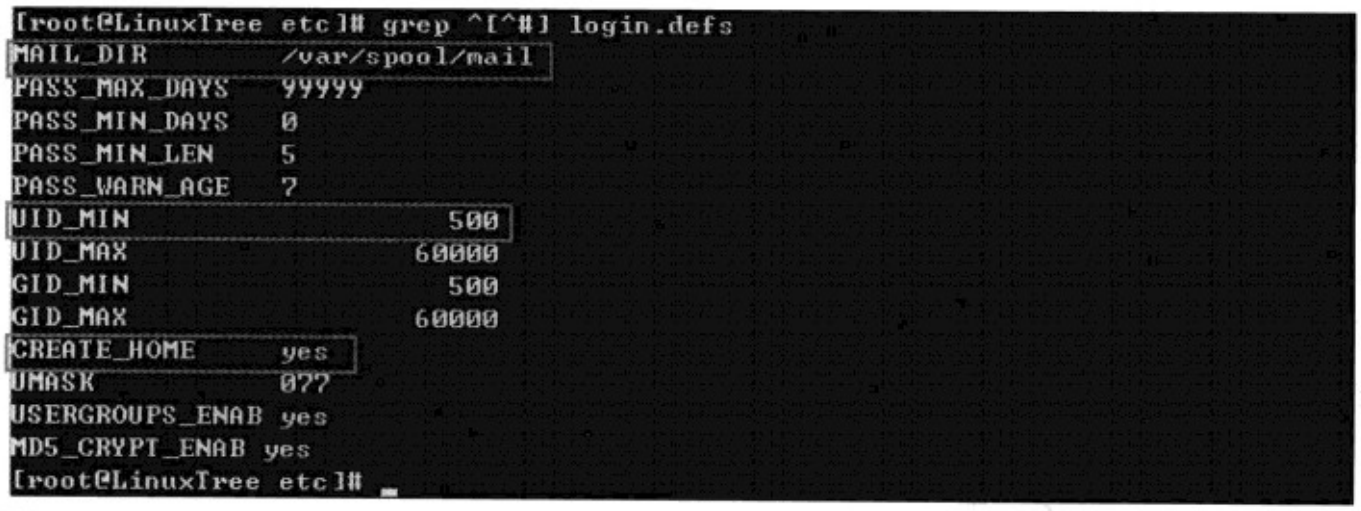


图 6-43：login.defs 文件的内容

◆ passwd

这个文件应该是所有系统管理员一开始学 Linux 就要知道的文件之一，因为这里面记录了所有系统用户的账号基本数据（如图 6-44 所示），每一个用户的记录是以下面的字段格式所组合而成的（以 newuser 为例），每一个字段以“:”隔开，总共有六栏：

用户名称	密码	UID	GID	全名	login shell
newuser	不存在此文件中	503	503	空白	/bin/bash

全名的部分可以自由填上，如果有填写（如图 6-44 中的“juergen”用户），当别的用户查询该账号，就可以在“Name”字段中显示其值，不过现在好像已经越来越没有用了（知道 finger 命令的人也不多吧）。

```
[root@LinuxTree etc]# tail -5 passwd
tclhttpd:x:500:500:Tclhttpd:/var/www/tclhttpd:/bin/false
gdm:x:42:42::/var/gdm:/sbin/nologin
juergen:x:501:501:Juergne Chiu:/home/juergen:/bin/bash
boa:x:502:48::/home/boa:/bin/bash
newuser:x:503:503::/home/newuser:/bin/bash
[root@LinuxTree etc]# finger juergen
Login: juergen                          Name: Juergne Chiu
Directory: /home/juergen                Shell: /bin/bash
On since Fri Jun 27 23:41 (CST) on pts/0 from 10.32.15.207
   13 minutes 41 seconds idle
On since Fri Jun 27 23:47 (CST) on pts/1 from 10.32.15.207
   1 second idle
Mail last read Mon May 12 09:39 2008 (CST)
No Plan.
[root@LinuxTree etc]#
```

图 6-44：passwd 文件的内容与影响

◆ shadow

因为在【/etc/passwd】文件中不能记录密码（容易遭窃取），因此 Linux 的作法是经过“hash”函数的处理后，储存在【/etc/shadow】文件中（如图 6-45 所示，图中有一长串的乱码，就是经过 hash 函数处理过的结果），提高了系统账号的安全性，所以该文件的重要性不言而喻。

```
[root@LinuxTree ~]# tail -3 /etc/shadow
juergen:$1$YwxQNxEm$j4l7M2qdP7GMNDIy0qN2C1:13939:0:99999:7:::
boa:!!:13987:0:99999:7:::
newuser:$1$ynXYKjXf$pFpSaHUivUk5WK98thYHi1:14057:0:99999:7:::
[root@LinuxTree ~]# _
```

图 6-45：shadow 文件中的部分内容

【/etc/shadow】文件的格式如表 6-5 所列，以图 6-45 中的【juergen】用户为例。

表 6-5：【/etc/shadow】文件的格式

编号	字段（由左至右）	值
1	账号	juergen
2	经处理过的密码	1YwxQNxEm$j4l7M2qdP7GMNDIy0qN2C1
3	密码从上一次更改的日期到 1970 年 1 月 1 日的天数	13939
4	几天内密码不可被变更	0
5	几天内需要重新改变密码	99999
6	密码过期的几天前要通知用户	7
7	密码过期的宽限天数	空白

续表

编号	字段（由左至右）	值
8	账号失效日起到 1970 年 1 月 1 日的天数	空白
9	保留字段	空白

6.1.2 服务器目录

在本节中，主要是将所有【/etc】目录中和“服务器架设相关”的子目录，整理在一起，方便读者阅读。

◆ BackupPC

这是一套专门针对 Linux、Windows 及 Mac 操作系统提供备份服务的软件（而且是多台计算机，不局限于几台，甚至可以给 DHCP 的网段下所有计算机做备份）。此外，它也是主要配置文件存在的地方，但其主目录却是在/usr/share/BackupPC 下，即使系统以外的服务器软件（如图 6-46 所示，一般的服务器软件是直接将所有文件放在“/”下的目录中的）。

```
[root@LinuxTree ~]# ls /etc/BackupPC/
config.pl  hosts
[root@LinuxTree ~]# ls /usr/share/BackupPC/
[root@LinuxTree ~]#
```

图 6-46：BackupPC 的配置文件及主要目录

这一套备份软件使用上非常简单，因为只有一个配置文件需要修改，就是【/etc/BackuPC】目录下的 config.pl 文件，而文件中的每一个配置参数都附有很详细的注释（如图 6-47 所示），所以，要将一台主机变为 BackupPC 的服务器，基本上不会太复杂，只须修改一份很清楚的配置文件即可。另一个方便之处就是它也提供网页接口给管理员配置、修改及监控的功能。

```
# Share name to backup.  For $Conf{XferMethod} = "rsync" this should
# be a file system path, eg '/' or '/home'.
#
# For $Conf{XferMethod} = "rsyncd" this should
# to backup (ie: the name from /etc/rsynd.conf
#
# This can also be a list of multiple file sys
# For example, by adding --one-file-system to
# can backup each file system separately, which makes restoring one
# bad file system easier.  In this case you would list all of the mount
# points:
#
#     $Conf{RsyncShareName} = ['/', '/var', '/data', '/boot'];
#
$Conf{RsyncShareName} = '/';
```

在一项设定参数前面，都会有一段非常详细的参数说明，让用户清楚了解每一项参数的目的与功能。

设定有哪些目录是要进行备份的

图 6-47：BackupPC 配置文件的部分内容

◆ boa

boa 是一个小巧的网页服务器程序，和一般所使用的 httpd（Apache 所使用的 daemon 名称）最大的不同在于，boa 没有用所谓的 fork( )函数，是单一进程的奋战。简单地说，httpd 在执行时会由一个父程序同时启动多个子程序，因此，在处理进入的 HTTP 连接时是由子程序去面对 HTTP 的用户（也就是网页程序的联机，像 IE），但也因此对系统的内存会有较大的需求；而 boa 则刚好相反，是由单一的进程处理所有进入的 HTTP 联机，因此占系统中内存较小的空间，对内存较小的系统很有用。目前很多嵌入式系统，都会以 boa 为默认的网页服务器软件。

如图 6-48 所示，httpd 与 boa 两种不同的网页服务器软件，在执行后的程序处理以及占用内存大小都不一样，boa 的目标在于速度（性能）与安全性（防止被恶意用户所破坏，并非是加密上的功能）上的提升，以不同一般流行的网页服务器（像 Apache）的角度为出发点。当 boa 的 daemon 在执行时，除了单一的程序处理外，也只会占用到内存极小的空间，非常轻巧（在图 6-48 为 512MB 中的 0%），有兴趣的读者可以在 boa 的官方网站上找到所需的信息（在图 6-49 所示），网址如下：

http://www.boa.org

图 6-48：httpd 在执行时所占用的内存

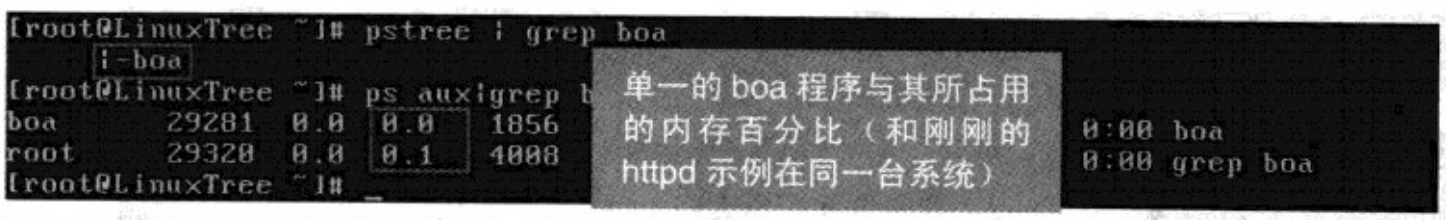


图 6-49：boa 在执行时所占用的内存

◆ cups

cups 全名为 common unix printing system，也就是 Linux 下的打印机服务器，所以此目录下都是这服务的配置文件，有兴趣在 Linux 下架设打印机服务器的读者，可以参考网络上的详细说明（使用 google 就可以查到一大堆），这里就不再列出参照网址，因为笔者从来都没有想过要使用

Linux 当作打印机服务器。理由其实很简单，因为不想在忙碌的时候再找一堆麻烦事，你是否可以想象在 Linux 下打印机出问题时，可以支持的厂商应该是很有限的（代理商或店面要有办法解决 Linux 下硬件问题的实在不多），还是使用“打印机所熟悉的 Windows”吧！

◆ dnsmasq.d

该目录是 dnsmasq 专用的配置文件目录，dnsmasq 是一种 DNS 的“轻薄机种”，专为区域或小型网络所设计，拥有比一般 DNS 更为方便简易的配置，另外，也贴心地为一些无盘系统的用户，将 DHCP、BOOTP 及 TFTP 的服务加在其中，不过当然也是“轻薄型”的。

dnsmasq 的主要配置文件是在【/etc】下的 dnsmasq.conf，其配置方式和以往的 DNS、DHCP 都不太一样，都是以少许的行数及较直观的配置方式就可以进行配置（如图 6-50 所示），所以在配置的方便性上是比较好的，只不过因为在同一份配置文件中包含 DNS、DHCP 及 TFTP 的配置，项目会有点乱。

```
[root@LinuxTree etc]# grep ^#dhcp dnsmasq.conf|head -5
#dhcp-range=192.168.0.50,192.168.0.150,12h
#dhcp-range=192.168.0.50,192.168.0.150,255.255.255.0,12h
#dhcp-range=red,192.168.0.50,192.168.0.150
#dhcp-host=11:22:33:44:55:66,192.168.0.60
#dhcp-host=11:22:33:44:55:66,fred
[root@LinuxTree etc]# _
```

比原本 DHCP 方便的设定方式

图 6-50：/etc/dnsmasq.conf 文件的部分内容

如果在该配置文件有尚未加入或需要额外配置的，就可以将其他配置项目建立在现在介绍的【/etc/dnsmasq.d】目录中。

1. exim

在 Mail 的使用上，分为 MTA（Mail Transfer Agent）、MDA（Mail Delivery Agent）与 MUA（Mail User Agent）三种角色。

在服务器端以 MTA 为主要的服务软件（像一般常听到的 Sendmail），用来协助内部或外部的用户收发 Email。

用户端为 MUA（像 Outlook），当作是 End User（终端用户）收发 Email 用的 Client 端软件，也最容易操作。

而 MDA 则存在于服务器端，并协助将收到的信件放置在正确的用户信件目录中（像 /var/spool/mail）。

而 exim 就是 MTA 的其中一种软件，一般在 Redhat 下默认是以 sendmail 这个软件为主要的 MTA 软件，exim 则是 Debian 默认所使用的 Mail Server 软件，但它在 Redhat 及 SuSE 上也都可

以使用。

2. httpd

这是 Linux 下最常用到的一个服务器目录，也就是网页服务器（默认为 Apache）的主要配置所在，在该目录中（如图 6-51 所示），其实主要只有三个子目录（conf、conf.d 和 modsecurity.d），分别储存不同目的的配置文件，最主要的配置文件在 conf 子目录下的 httpd.conf。而其他子目录（logs、modules 和 run）都只以链接方式存在，方便用户链接到适当的路径。

```
[root@LinuxTree httpd]# ls
conf  logs  modules  run
[root@LinuxTree httpd]# _
```

图 6-51：/etc/httpd 目录下的列表

3. lighttpd

相对于上一个 httpd 的目录，lighttpd 也是网页服务器所使用的配置目录，只是由不同的组件所提供。lighttpd 和 httpd 的主要差别，在于 lighttpd 属于轻巧型的网页服务器软件，它的目标是要以“较少的 CPU”及“较少的内存”来达到“较高的性能”，不过这一类软件（轻量级）当然在功能上就不会像原本的 httpd 有各式各样的配置可使用，但仍然有许多特点，读者可以参考以下的网址或 wiki 取得最新的信息。

http://www.lighttpd.net/

4. mail

该目录对要架设 Mail Server 的用户来说是非知道不可的，因为这是 Redhat 默认 Mail Server 组件【sendmail】的主要配置目录，在比较早期的操作系统中，该目录名称是【sendmail】，后来才换掉。在该目录中，除了主要的 Mail Server 配置文件外，也包含反垃圾邮件的 SpamAssassin 组件的配置，使对邮件过滤的配置在同一目录中就可以搞定。

5. news

这目录是 Linux News Server 的主要配置目录，也由 INN（Internet News）软件所提供，如果想要架设 News Server，到这个目录就对了，不过，现在网络的论坛及 google 之风行，会用 News 软件的人更少了，因此 News 的使用量已经大幅降低，可用性自然也越来越低了。

6. ntp

Network Time Protocol 服务主要的配置目录，不过，最主要的 NTP 配置文件却是放在【/etc】

下的 ntp.conf 文件，在【/etc/ntp】目录下主要是保留一份“keys”文件，用来让对时系统有一个认证的机制，其他主要的配置，包含对时主机及权限的控制，全部都在【/etc/ntp.conf】中。

7. openldap

本目录很明显是 LDAP（Lightweight Directory Access Protocol）的配置目录，只不过这是针对 OpenLDAP 软件的配置目录（OpenLDAP 只是 OpenLDAP 组织遵循 LDAP 协议所存在的一个软件），有需要详细的操作步骤或配置方式的，可以参考以下网址：

http://www.tldp.org/HOWTO/LDAP-HOWTO/

8. postfix

这是 Postfix 组件所提供的主要配置文件目录，该软件是继 Sendmail 之后很流行的一种 Mail Server 软件，和 Sendmail 最大的差异，就在于它的配置文件【main.cf】非常简单（相对于 Sendmail 的主要配置文件 sendmail.cf），其支持的内容大都和 Sendmail 一样，算是 Sendmail 的修正版。但请不要误会，Postfix 是完全重写的，所以里面的程序代码和 Sendmail 完全不一样，只是当初 Postfix 的作者希望可以和 Sendmail 兼容，让用户在转换上简单一点，故在整体的设计上和 Sendmail 很相像。

9. pulse

这是一个 PulseAudio 的主配置目录，它是一个网络上传送声音的服务软件，有点像是声音的 Proxy Server，通过这一个软件，可以在传送声音的过程中，进行一些声音的处理，像是增加声音之类的操作。这个软件有一个很大的优点，就是提供非常多现成的插件（plugin modules）供用户使用，因此，它很适合一些提供声音的公司（像广播公司、新闻媒体或是音乐相关网站）使用，有兴趣的读者可以到该公司的官方网站，参考他们所提供的文件或模块。

http://pulseaudio.org/wiki/AboutPulseAudio

10. samba

samba 是在 Linux 下非常有名的文件共享服务，主要的配置文件就在【/etc/samba】下的 smb.conf，配置文件的格式算是简单明了，所以大部分想要在 Linux 下和微软的网上邻居互传数据时，都会想到这一个服务，而且如果不论权限问题，要共享出一个目录真的非常快（其实每一种服务，只要无须高级的配置，好像都可以很快地产生出来）。samba 也可以加入微软的 AD 中，在用户上可以利用该目录下的【smbusers】文件记录 Linux 与 Windows 下账号的转换方式，不过，说得容易做得难，很多兼容性的问题看来是要一段时间才有机会解决的，方法有了，但兼容性的

问题却始终存在。

11. smrsh

这是 Sendmail 为了限制用户可使用的命令所设计的程序，将原本用户所使用的【/bin/sh】替换为【/usr/sbin/smrsh】（发送 mail 时的 shell，不是登录时的环境）。在这个目录中，可以存放一些希望让用户使用的命令（像是 Sendmail 外其他 mail 寄送用的命令），默认是空的，但要切记放在这目录中的程序，将是让别人有机会通过 mail 的服务直接运行的命令，等于多开了一个危险的后门，因为不用登录系统也可以使用，所以要特别小心。

12. snmp

SNMP（Simple Network Management Protocol）是一套简易的网络管理协议，一般在网络设备上都会支持这一项协议，非常普遍，大部分只要是流量统计、硬件资源统计之类的数据或报表，都是由 SNMP 的协议所提供的数据产生的。在目录中唯一的一个文件【snmpd.conf】是该服务的主配置文件，一般不需要用到这个文件（一般系统只要将 snmpd 的 daemon 启动就已经可以使用 SNMP 的服务），大部分需要修改的是 Client 端统计软件的配置文件，像之前很有名的 MRTG 就是利用 SNMP 的数据进行分析的。现在越来越多的统计软件出现，使用上也越来越简单了。

13. squid

这就是 Linux 下的代理服务器——Squid 的配置文件目录，主配置文件为【squid.conf】文件，大部分的配置都在该文件中，当然除了这个之外，还有几个独立的配置文件，像是专门在记录所有网络上文件格式的“mime.conf”、用来整合 NT 网域认证机制的“msntauth.conf”（刚看到时还以为是 MSN 用的配置文件……想用 MSN 想疯了吧！）或是管理 cache 的“cachemgr.conf”。

14. ssh

这是 SSH 服务的主要配置目录，最主要的配置文件为【sshd_config】，除了配置文件外，还存放一些 SSH 联机时所需使用到不同算法及版本的公钥（Public Key）和私钥（Private Key）。如果用户在使用时，还是必须以主目录下的【.ssh】为主要密钥存放的目录。另外要注意的一点，目前系统安全性默认都只可以让 root 做到用 ssh 免认证的方式登录系统（当然要有一些事前的密钥配置），普通用户是不被允许的。

15. tclhttpd

这又是另外一种 Web Server 的软件，由 Tcl Developer Xchange 组织所提供，主要的特点是用

TCL 程序语言（请念成 tickle）所开发的，这个【/etc/tclhttpd】的目录就是它主要配置文件所在的地方。当然，一般 Web Server 的功能一定都会有（不然谁要用），除此之外，还有一个重要的因素，就是这一个软件，可以让本身嵌入（embedded）在另外一个软件中，让另一套软件具有提供 Web 的特性。其他比较技术性的部分，请直接参考该官方网站中有关“Advanced Features”的介绍。

http://www.tcl.tk/software/tclhttpd/

16. vsftpd

在 Linux 下默认的 FTP 服务器软件很多都是以 vsftp 为主的，因为比之前的 wu-ftp 安全性要高，所以大部分的系统都已转到 vsftp 软件，该目录是所有 vsftp 配置相关的文件所在，当然也包括主配置文件【vsftpd.conf】文件，除此之外，也有限制用户的【user_list】列表，但也要注意 PAM 机制对 vsftpd 的机制意味着什么，不然就算 vsftp 的配置再成功也没有用。

17. xinetd.d

【xinetd】是一个管理多个服务的 daemon，在它下面可以有多个服务，所以这一个目录中所存放的文件都是一些依附在【xinetd】daemon 之下的服务程序，像最常看到的【telnet】就是最明显的例子。除了这些在目录中的服务之外，有些以往都是用【standalone】模式（即自己本身是独立的 daemon）的服务，有些其实也可以通过配置，将其服务的 daemon 归属在【xinetd】之下，像前面才提到的【vsftpd】的配置文件中“listen”参数，就是用来决定是否要让【vsftpd】以 standalone 模式或是依附在 xinetd 之下的参数。

如图 6-52 所示，经过笔者对 vsftpd 的配置，可以将 vsftpd 服务交由 xinetd 托管，当用户在使用 telnet 或 vsftpd 时，在程序上看到的都是由 xinetd 一并管理，而不是由 telnet 或 vsftpd 自己启动的。

```
[root@LinuxTree etc]# pstree|tail -3
      |-xinetd-+-2*[in.telnetd---login---bash]
      |        `-vsftpd---vsftpd
      `-yum-updatesd
[root@LinuxTree etc]#
```

图 6-52：xinetd 所管理服务的状态

在【/etc/xinetd.conf】文件中的配置，是针对所有依附在 xinetd 之下的服务配置的，但在【/etc/xinetd.d】目录中所有服务的配置文件（如图 6-53 所示），就是针对每一个服务的个别配置，在使用上要特别注意。至于如何使用 xinetd.conf 或是每一个服务的配置文件，可以参考 Linux 内附 xinetd.conf 的 man page，里面已经清楚记载了所有参数代表的意义，读者会发现交给 xinetd 管还真是方便。

```
[root@LinuxTree xinetd.d]# ls
chargen-dgram   daytime-dgram   discard-stream  rsync          tftp
chargen-stream  daytime-stream  echo-dgram      tcpmux-server  time-dgram
cvs             discard-dgram   echo-stream     telnet         time-stream
[root@LinuxTree xinetd.d]#
```

图 6-53：/etc/xinetd.d 目录下的文件

6.1.3 系统目录

系统目录主要都是将一些在【/etc】目录下和系统运行相关的子目录收纳进来。

◆ blkid

本目录中所存放的其实是一个块设备 ID（Block ID）的临时文件（如图 6-54 所示），主要是记录（但只是暂存的，也就是当执行 blkid 命令时，会直接被更新为新的值）系统中所有区块设备的标签名称（Label Name）、硬件的唯一识别码（UUID 值）、文件系统的格式等基本信息。

```
[root@LinuxTree blkid]# blkid /dev/sda1
/dev/sda1: LABEL="/boot" UUID="14ac957f-8c87-4c2c-a18e-196233ba0bcf" SEC_TYPE="e
xt2" TYPE="ext3"
[root@LinuxTree blkid]# cat blkid.tab
<device DEVNO="0xfd01" TIME="1208537054" TYPE=  0-LogV
ol101</device>
<device DEVNO="0xfd00" TIME="1208537054" UUID=  a6e9d8
e00" SEC_TYPE="ext2" TYPE="ext3">/dev/mapper/
<device DEVNO="0x0801" TIME="1208537063" LABEL="/boot" UUID="14ac957f-8c87-4c2c-
a18e-196233ba0bcf" SEC_TYPE="ext2" TYPE="ext3">/dev/sda1</device>
<device DEVNO="0xfd01" TIME="1208537054" TYPE="swap">/dev/VolGroup00/LogVol01</d
evice>
<device DEVNO="0x0811" TIME="12085  5b935e7ea
33e" TYPE="ext2">/dev/sdb1</device
<device DEVNO="0x0b00" TIME="12032  9660">/de
v/cdrom</device>
<device DEVNO="0xfd00" TIME="12085  1e8a6e9d8
e00" SEC_TYPE="ext2" TYPE="ext3">/dev/root</device>
<device DEVNO="0x0b00" TIME="1203592536" LABEL="FC/6 i386 DVD" TYPE="iso9660">/d
ev/cdrom-sr0</device>
[root@LinuxTree blkid]#
```

在 blkid 执行后，如果有不一样结果，会将/etc/blkid/blkid.tab 文件中的值替换掉

在 blkid.tab 文件中，有所有区块设备的相关信息，但这些信息都是通过 blkid 所产生出的临时文件，每次都会被更新

图 6-54：blkid 命令的结果与其暂存盘的对照图

◆ bluetooth

在 Linux 下使用蓝牙设备所需的配置文件，启动蓝牙检测的主要服务仍是【/etc/rc.d/init.d/bluetooth】，该程序会参考本目录中的“hcid.conf”主配置文件，或其他的额外配置文件。

◆ cron.X

cron.X 的目录都是给 Cron 软件存放其需要任务计划的文件所使用的，按任务计划时间的长短及配置特性分为 cron.d、cron.daily、cron.hourly、cron.monthly、cron.weekly 五个主要目录，让用户可以按想要的任务计划方式，放在正确的目录中，但因为这些目录都同属于 cron 机制下的目录，所以全部归类到【cron.X】中。

除了 cron.d 的目录外，其他的 cron 目录都是由/etc/crontab 文件所控制的（如图 6-55 所示），对用户来说，只要将文件放入以分、时、日、星期、月为单位的目录，就可以由 cron 自动为用户执行，但缺点就是无法再自定更细的时间，都是以/etc/crontab 文件中所配置的时间为主。

图 6-55：/etc/crontab 的内容

- /etc/cron.d：这个目录是可以按用户的本身需求所存放的目录，为什么这样说，因为在/etc/cron.d 目录中，用户可以自定义其执行的周期，也就是说其中的文件都是可以自定义的，弹性自然比较大（如图 6-56 所示）。

图 6-56：/etc/cron.d 目录中的示例文件

- /etc/cron.hourly：以一个小时为单位，用户若将所需执行的文件放到这个目录下，就会在“每小时零一分”执行该文件（默认值），方便在于不用再配置任何文件，只要放到该目录下即可。
- /etc/cron.daily：以日为单位，只要放在该目录下的文件都会在“每天的 4 点 02 分”（默认值）执行一次。
- /etc/cron.weekly：以星期为单位，也就是说是以一个星期的“星期几”要执行为基本单

位，只要放在该目录下的文件都会在“每周日的 4 点 22 分”（默认值）执行一次。

- /etc/cron.monthly：以月为单位，只要放在该目录下的文件都会在“每月 1 号的 4 点 42 分”（默认值）执行一次。

◆ dbus-1

D-BUS 的主要配置目录，还记得在“第 3.2.11 节：/proc/sysvipc”中提到的 IPC（Interprocess Communication）信息交流的方式吗？D-BUS 也是一种 IPC 交流的方式。和传统 IPC 的差别在于，IPC 的主要功能是软件经由“monolithic process”（集合程序）的程序处理技术，会成为单一线程（single thread）的执行方式，也就是当某程序在执行时，会以一个线程的方式交给 CPU 处理，即以一个应用程序为基本单位。所以，交互上会是“应用程序对应用程序”的方式，这对于目前程序越来越多且复杂度也越来越高的系统而言，已经不是很好的解决方案了。

而 D-BUS 是一种 IPC 改良后的交互方式，D-BUS 其信息的传送方式由原本的以应用程序为单位，转变为以对象为基本的交互单位，因为每一个应用程序都可以自定义其对象的多少，因此，在“对象至对象”的交互方法上，D-BUS 又增加了“路由”概念，让每一个对象可以正确地找到另一个对象，而这个路由的工作，就是交给【dbus-daemon】的服务来处理。所以 D-BUS 的机制，对于程序设计师及需要程序间彼此交互的开发者来说，是一个很重要的管道。

有兴趣的读者可以参考 Redhat 官方网站的说明：

http://www.redhat.com/magazine/003jan05/features/dbus/

◆ default

这里是存放一些系统软件默认值的目录，存放某些软件执行时的基本参数，默认里面会有 useradd 命令的基本信息（如图 6-57 所示），从这个文件中就可以知道新增用户时，会帮用户所产生的目录路径、Shell 种类、默认来源目录等信息。当然还有其他的文件，每一个文件都代表一个软件所执行时的默认条件，在使用软件上很方便，可以取代一些常需使用的参数。

```
[root@LinuxTree default]# pwd
/etc/default
[root@LinuxTree default]# cat useradd
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes

[root@LinuxTree default]#
```

当新增用户时的基本参数

图 6-57：/etc/default 目录中 useradd 文件的内容

◆ fedora

这是一个有关 Fedora 系统用户管理的目录，也和之前提到的/etc/alternatives 目录有直接关系，因为该目录下所存放的，正是 Fedora 希望新增或删除用户时所遵循的标准（也就是像 useradd 或 groupadd 之类的命令）。不过，因为 fedora 这一个目录目前只能算是建议使用的阶段，并没有要求软件开始采用，在用户与用户组的管理上，其实和这个目录没有关系，除非将这些相关软件都改为以 Alternative 的方式执行，如图 6-58 所示。

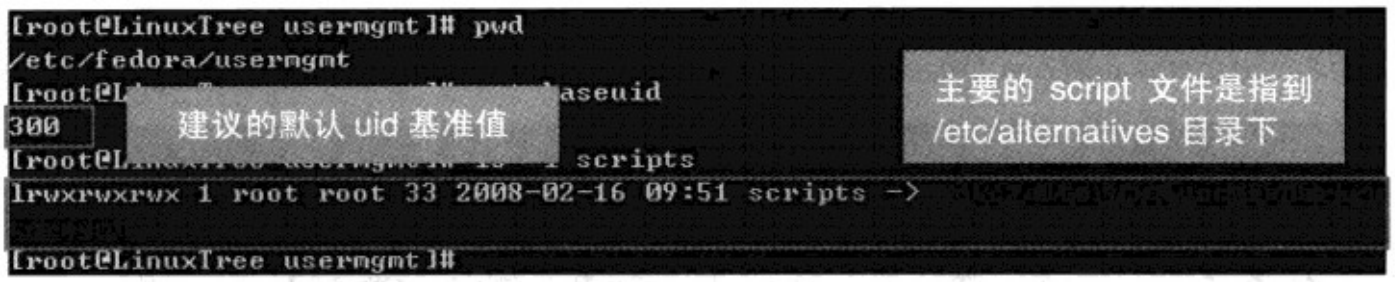


图 6-58：/etc/fedora 目录下的某些文件

不过/etc/fedora 目录下的文件，虽然也是用户管理的组件所提供的，但和默认的用户管理组件完全不同。默认的组件是由 shadow-utils 所提供的，而/etc/fedora 下的文件则是由 fedora-usermgmt-shadow-utils 所产生的（如图 6-59 所示），两者主要会用到的命令路径完全不一样，因此在未来虽然有可能会换成 fedora-usermgmt-shadow-utils 组件的管理方式，但目前是没有影响的。

```
[root@LinuxTree usermgmt]# rpm -ql shadow-utils|grep /usr/sbin/useradd
/usr/sbin/useradd
[root@LinuxTree usermgmt]# rpm -ql fedora-usermgmt-shadow-utils|grep useradd
/etc/fedora/usermgmt/scripts.legacy/useradd
/etc/fedora/usermgmt/scripts.shadow-utils/useradd
[root@LinuxTree usermgmt]#
```

图 6-59：shadow-utils 组件与 fedora-usermgmt-shadow-utils 的一些差异

◆ firmware

这个目录所存放的东西是非常底层的信息，是 CPU 所需的 microcode 的实体文件。所谓 microcode，是当用户在执行程序时，由高级语言（像 C）转变为 CPU 命令（instructions）过程中间的一种“方式”，所以在 microcode 中定义了许多 CPU 的处理器行为。

一般来说，microcode 是在用户拿到主板的时候，就已经在主板上的 BIOS 中了，所有更新也都是由 BIOS 更新（简单地说就是更新 BIOS）的，但因为有时候通过 BIOS 的更新可能会有管理上问题（最基本的就是一定要重启），在 Linux 及部分系统提供更新 microcode 的机制，即【microcode_ctl】组件（Linux 下的名称）。

从图 6-60 中可以看出，这一份 microcode 实体文件是以文字文件的方式存在，且在这份文件中，应该已经包含所有 CPU（视该文件所支持的程度）最新的 microcode 定义。Linux 的 microcode 更新机制就是靠 firmware 的目录，在某些情况下，可能启动时会出现 microcode error 之类的问题，通常只要将 firmware 目录下的这个文件更新就可以解决了，如果系统没有提供适当的 microcode 文件，也可以直接从下面的 Intel 网站上下载最新的 microcode 文件，AMD 则是到目前为止似乎没有提供更新 microcode 的方式。

Intel Microcode 下载网址：

http://downloadcenter.intel.com/Detail_Desc.aspx?ProductID=1077&DwnldID=14303&lang=zht

```
[root@LinuxTree firmware]# pwd
/etc/firmware
[root@LinuxTree firmware]# ls
microcode.dat
[root@LinuxTree firmware]# tail -5 microcode.dat
0xb37760b1,     0x9665d6fa,     0xb27ec974,     0xffa844d9,
0xc2cd368f,     0xd608dd3c,     0x0144c468,     0xff855bda,
0x38b722ab,     0xe0d03d11,     0x481f7936,     0xe333cebc,
0xab099835,     0x4916bd84,     0x94976044,     0x7d444113,
0xfd7138f3,     0x3aab1781,     0x353ee8c2,     0x3f3b258c,
[root@LinuxTree firmware]#
```

图 6-60：/etc/firmware 目录下文件的内容

◆ foomatic

用户在使用打印机的时候，除了单纯的一对一的连接方式外，另外有可能的方式就是一对多（单机接多台打印机）或多对一（多台 PC 接一台打印机）的环境。多对一最常看到的解决方案就是用 Spool（提供打印数据的缓冲区）的方式解决（像 CUPS 的打印机服务）；而一对多就是由这个【foomagic】目录所提供的方式来克服的，不然每次只要换一台打印机打印，就需要将所有的配置再来一次，太麻烦。

在 foomagic 中，可以记录多条打印机数据，让用户只在使用前先行配置所有需要使用的打印机即可，如此一来就能在打印时直接选取需要的打印机，正确打印。

◆ hal

HAL 全名为 Hardware Abstraction Layer，是 Linux 一种管理硬件的机制（如图 6-61 所示），它会帮所有的应用程序或用户搜集所有 PCI 及 USB 等硬件信息，而 Linux 底层的一些硬件检测机制（像 udev、hotplug 等）会提供 HAL 相关的硬件信息，因此，用户可以很简单并实时地通过 HAL 的方式取得硬件的相关数据。

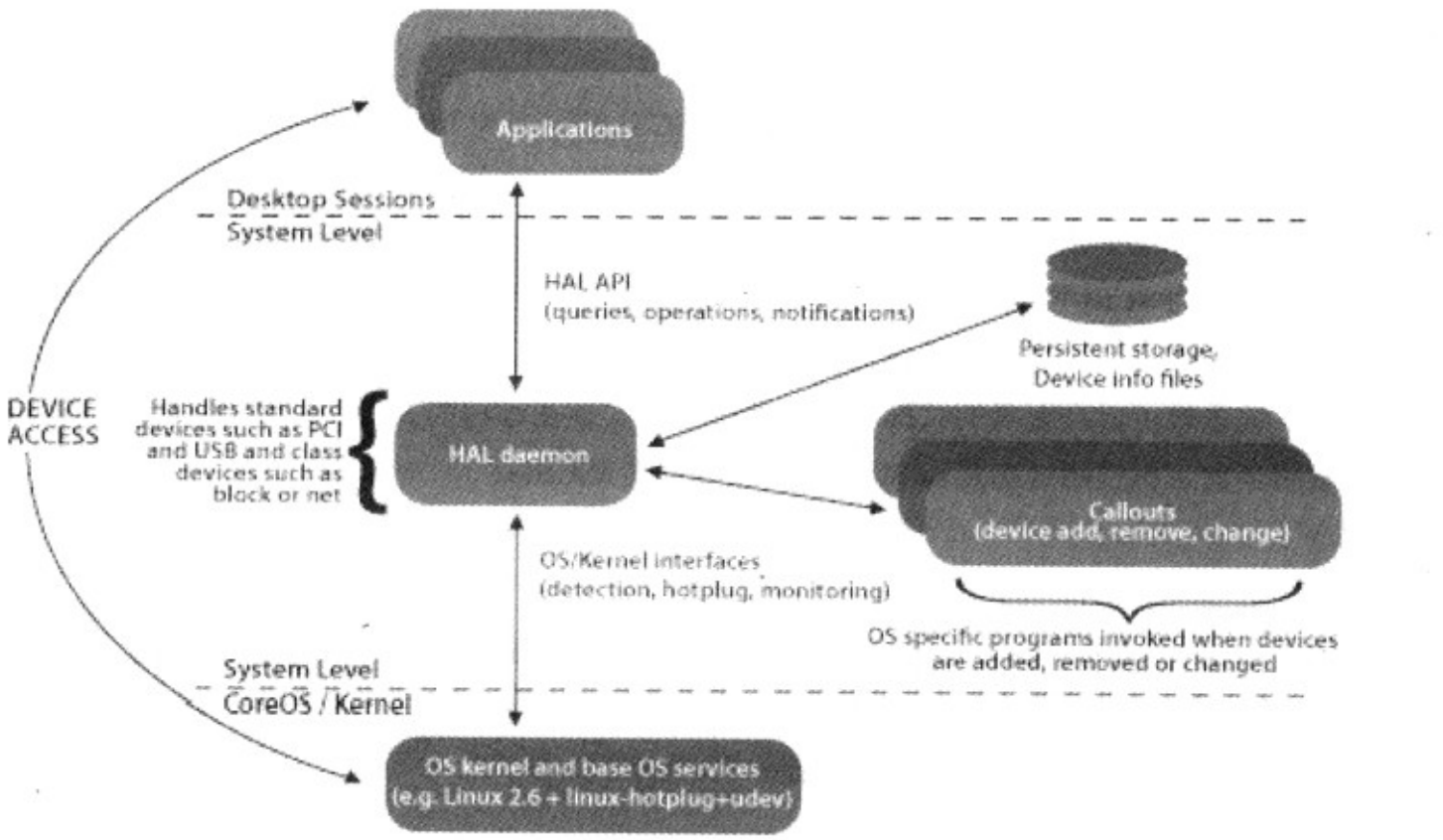


图 6-61：HAL 的架构图[1]

原本目录应该存放在 HAL 相关的硬件配置、设备列表、规则文件中。然而，目前看起来所有 HAL 的信息似乎并没有真的放在【/etc/hal】的目录下（如图 6-62 所示），笔者所看到其中的目录都是空的，而真正所需要的硬件相关文件，Redhat 都另外存放在【/usr/share/hal】的目录下，当用户要管理 HAL 时，请注意目录的差异。

```
[root@LinuxTree hal]# find /usr/share/hal/|wc -l
60
[root@LinuxTree hal]# find /etc/hal/|wc -l
5
[root@LinuxTree hal]#
```

图 6-62：/etc/hal 目录与/usr/share/hal 目录的文件数目比较

◆ hp

从目录名称上就可以知道，目录是为了 HP 公司的产品所存在的，但实在搞不懂 Redhat 为何会为 HP 在/etc 下默认建立这一目录，因为这是专门让 HP 放置有关图片或文件的打印、扫描及传真的配置（HPLIP，HP Linux Imaging and Printing），并非所有用户都适用的系统配置，或许是因为 HP 的产品太通用。但这个目录下只有一个配置文件，其他主要的执行文件或其他文件，都被放到了【/usr/share/hplip】的目录中。

[1] 图片来源：*http://www.redhat.com/magazine/003jan05/features/hal/*。

◆ iscsi

目录是 iSCSI 协议的主要配置文件存放区，iSCSI 的全名为 Internet SCSI，它的主要的功能是通过 TCP/IP 的协议传送 SCSI 的数据包，虽然听起来有点奇怪，但实际上它的目的就是如此。iSCSI 可以让系统通过网络的架构，直接连接到网络上的某一台 Storage（储存设备），因为传送中的数据包内部为 SCSI 协议，因此，可以将该 Storage 直接变为系统上的某一块 SCSI 硬盘，而不是像 NFS、SAMBA 之类的，只能做分享目录使用，或将远程的 Storage 模拟为实际的硬盘，也就是说，该系统可以在无盘的状况下，将操作系统安装在远程的 Storage 上，并且一台 Storage 可以共享给多台主机使用（当然该 Storage 一定要支持 iSCSI）。

◆ isdn

ISDN（Integrated Services Digital Network）服务的主要配置目录，里面包含可拨号的用户、电话、联机方式等，现在的大部分 ISDN 都拿来当作一种备用的方案，或者是语音上高音质的一种传输方式，至于一般的网络使用，已经是一种较旧的机制了。

◆ ld.so.conf.d

这个目录是 ldconfig 所使用的，更准确地说，它是由/etc/ld.so.conf 文件所决定的（如图 6-63 所示），ldconfig 命令的目的，在于将系统中的一些函数库预先存放到内存中（当然是挑一些比较常用的），让系统在使用时（读取）可以比以往通过硬盘的读取速度来得快，这样可大幅增加系统的性能，尤其当要重复读取时更明显。

ldconfig 要将哪些函数库丢到内存中，则须看/etc/ld.so.conf 文件中所记录的信息，从图 6-63 中被圈选的内容可以清楚地知道，它所指向的正是【/etc/ld.so.conf.d】目录，因此，这代表只要在该目录下所配置的所有目录配置，均可以通过 ldconfig 命令，全部加载到内存中。

```
[root@LinuxTree ld.so.conf.d]# cat /etc/ld.so.conf
include ld.so.conf.d/*.conf
[root@LinuxTree ld.so.conf.d]# ls
ctapi-i386.conf                    mysql-i386.conf
kernelcap-2.6.20-2925.9.fc7.conf   qt-i386.conf
[root@LinuxTree ld.so.conf.d]#
```

图 6-63：/etc/ld.so.conf 文件与 ld.so.conf.d 目录的关系

如图 6-63 中的目录，有一个【mysql-i386.conf】文件，其中记录了 mysql 所须预先放在内存中的函数库路径（/usr/lib/mysql），在【ldconfig】命令执行完成后，就可以通过【ldconfig -p】命令找到已加载的 mysql 相关函数库，通过显示的信息，可以确定所加载的函数库就是【ld.so.conf.d】下所配置的函数库路径，如图 6-64 所示。

```
[root@LinuxTree ld.so.conf.d]# ldconfig -p|grep mysql
        libmysqlclient_r.so.15 (libc6) => /usr/lib/mysql/libmysqlclient_r.so.15
        libmysqlclient.so.15 (libc6) => /usr/lib/mysql/libmysqlclient.so.15
[root@LinuxTree ld.so.conf.d]# cat mysql-i386.conf
/usr/lib/mysql
[root@LinuxTree ld.so.conf.d]# ls /usr/lib/mysql/
libmysqlclient_r.so.15      libmysqlclient.so.15      mysqlbug
libmysqlclient_r.so.15.0.0  libmysqlclient.so.15.0.0  mysql_config
[root@LinuxTree ld.so.conf.d]#
```

图 6-64：已加载内存中的 mysql 函数库

◆ logrotate.d

目录对系统管理员来说，是十分重要的一个目录，因为目录中的文件，记录了如何定期备份系统所须要备份的系统或软件记录文件及备份方式。目录是由 logrotate 组件所提供的，而里面所有文件（如图 6-65 所示）是由各软件各自产生的，如果希望记录文件由 logrotate 来管理，请自行将其 logrotate 的配置文件放在这个目录下，logrotate 程序就可以通过 Cron Table 的方式，每天按时进行记录文件的备份工作，至于如何备份或备份数量为多少，可交由各需要备份的软件自行在各自的配置文件中做细节配置。

```
[root@LinuxTree logrotate.d]# ls
BackupPC    httpd      named    samba           squid    tux
bittorrent  kdm        ppp      sa-update       syslog   vsftpd.log
boa         lighttpd   psacct   setroubleshoot  thttpd   yum
exim        mgetty     rpm      snmpd           tomcat5
[root@LinuxTree logrotate.d]#
```

图 6-65：/etc/logrotate.d 目录中的文件列表

在【/etc/logrotate.conf】配置文件中，有一些针对 logrotate 备份操作的基本配置，如备份文件的数量默认值为“4”，就是在这里配置的，【/var/log】目录下的记录文件，都只会进行到第 4 个备份就停止（如图 6-66 所示），过期的会被删除。

```
[root@LinuxTree logrotate.d]# ls /var/log/*.[1-4]
/var/log/boot.log.1   /var/log/maillog.4   /var/log/secure.3
/var/log/boot.log.2   /var/log/messages.1  /var/log/secure.4
/var/log/boot.log.3   /var/log/messages.2  /var/log/spooler.1
/var/log/boot.log.4   /var/log/messages.3  /var/log/spooler.2
/var/log/cron.1       /var/log/messages.4  /var/log/spooler.3
/var/log/cron.2       /var/log/rpmpkgs.1   /var/log/spooler.4
/var/log/cron.3       /var/log/rpmpkgs.2   /var/log/vsftpd.log.1
/var/log/cron.4       /var/log/rpmpkgs.3   /var/log/vsftpd.log.2
/var/log/maillog.1    /var/log/rpmpkgs.4   /var/log/vsftpd.log.3
/var/log/maillog.2    /var/log/secure.1    /var/log/wtmp.1
/var/log/maillog.3    /var/log/secure.2
[root@LinuxTree logrotate.d]#
```

图 6-66：/var/log 目录下的备份文件

因此，读者要分清楚，【/etc/logrotate.conf】配置文件为 logrotate 所需的基本配置；而这里介绍的【/etc/logrotate.d】目录配置文件，则是各软件自行决定的备份操作，如可以预定在备份完成后，将服务重新启动或执行某 script file 之类的事情。

◆ logwatch

上一个目录已介绍完如何备份记录文件，这一个目录就是记载如何分析记录文件并告诉用户的软件【logwatch】的配置目录。然而，虽然在【/etc/logwatch】目录中已经有现成的子目录，但 logwatch 机制使用的所有配置及执行，其实是【/usr/share/logwatch】目录下的文件（如图 6-67 所示），所以，目前【/etc/logwatch】仍属于一个空壳子，所有要配置如何分析及告知用户的配置，都必须在图 6-67 的【logwatch.pl】文件中做配置。

```
[root@LinuxTree logwatch]# ls

[root@LinuxTree logwatch]# ls -l /etc/cron.daily/0logwatch
lrwxrwxrwx 1 root root 45 2008-02-16 09:42 /etc/cron.daily/0logwatch -> ../..//u
sr/share/logwatch/scripts/logwatch.pl
[root@LinuxTree logwatch]#
```

图 6-67：logwatch 的实际配置文件位置

◆ lsb-release.d

LSB 是 Linux Standard Base 的简写，是一个由很多人所执行的项目，其目的是将所有的 Linux Distribution（就是发行 Linux 的厂商，像 Redhat、SuSE、Slackware 等等）定义为一些共同的标准，因为目前各版本都会有很多文件路径、使用方式及许许多多不同点，当各软件厂商开发程序时，如果要在各 Linux Distribution 上使用，则须要增加许多的麻烦（因为要先对每一个 Distribution 熟悉，才有办法让组件配合该 Distribution），因此，这个项目希望可以定义一套标准，让各软件厂商支持。目前目录中放置了一些空文件名以表示目前的 LSB 版本，需要利用 LSB 开发软件的可以参考 LSB 官方网站：*http://www.linux-foundation.org/en/LSB*。

◆ lvm

LVM 的全名为 Logical Volume Manager，目前安装所有 Redhat 的版本，默认在硬盘的模式都会以 LVM 的方式安装，因为这样对以后的空间使用弹性较大，但当出现问题的时候会比较烦。LVM 主要通过/dev/mapper 下的设备文件运行，这个目录是 LVM 的基本配置文件，但配置或操作一般都只须要通过 LVM 提供的命令，而不会用到这个目录，除非要使用到很高级的配置才会更改此文件。

◆ makedev.d

MAKEDEV 组件使用的为配置文件目录，MAKEDEV 组件主要用来产生设备文件，也就是说，在/dev 目录下的文件都由这个命令所产生，但一般大家比较常使用的新增设备文件的命令可能是 mknod，这两个的差别在哪？

MAKEDEV 命令可以产生所有/dev 下所需的设备文件，但它如何知道每一个硬件的属性及种类，须通过现在介绍的【/etc/makedev.d】目录下的“所有文件”来判断（如图 6-68 所示），也就是当建立某一个设备文件时（如图 6-68 中的 ttyS0），会先参考【/etc/makedev.d】目录中针对该设备文件的定义或属性。换句话说，当没有定义该设备文件时，就无法产生该设备文件。

```
[root@LinuxTree makedev.d]# ls -l /dev/ttyS*
crw-rw---- 1 root uucp 4, 64 2008-05-06 01:34 /dev/ttyS0
crw-rw---- 1 root uucp 4, 65 2008-05-06 01:34 /dev/ttyS1
crw-rw---- 1 root uucp 4, 66 2008-05-06 01:34 /dev/ttyS2
crw-rw---- 1 root uucp 4, 67 2008-05-06 01:34 /dev/ttyS3
[root@LinuxTree makedev.d]# grep ttyS.. 01linux-2.6.x
c $SERIAL              4  64  1 192 ttyS%d
c $SERIAL            154   0  1 256 ttySR%d
c $SERIAL            156   0  1 256 ttySR%d 256
c $SERIAL            174   0  1  16 ttySI%d
c $SERIAL            204   5  1   3 ttySA%d 0
c $SERIAL            204   8  1   4 ttySC%d 0
c $TTY               204  40  1   1 ttySG0
c $SERIAL            204  41  1   3 ttySMX%d
c $SERIAL            204 116  1  32 ttySIOC%d
[root@LinuxTree makedev.d]#
```

图 6-68：/etc/makedev.d 目录对设备文件的意义

mknod 是一个较单一的命令，它只负责产生有关区块或字符属性的文件，但既然有了 MAKEDEV，为何还要 mknod 呢？因为刚刚有提过，如果在【/etc/makedev.d】目录中缺少某些 MAKEDEV 所需的信息，则无法产生该设备文件（如图 6-69 所示），因此，只好依赖 mknod。但是要由 mknod 产生设备文件，首先，必须知道该设备文件所需的 Major 及 Minor ID（设备文件所使用的分类方式），以及设备文件的种类，但即使产生了设备文件，还是需要将其权限及所有者再自行修改，通过 mknod 产生设备文件的过程是比较复杂的，只有在【/etc/makedev.d】目录没有定义的情况下，mknod 才是比较快的做法。

```
[root@LinuxTree makedev.d]# MAKEDEV /dev/ttySa
don't know how to make device "ttySa"
[root@LinuxTree makedev.d]# mknod /dev/ttySa c 4 68
[root@LinuxTree makedev.d]# ls -l /dev/ttySa
crw-r--r-- 1 root root 4, 68 2008-05-06 15:08 /dev/ttySa
[root@LinuxTree makedev.d]#
```

图 6-69：通过 mknod 新增设备文件的原因

◆ modprobe.d

目录是 modprobe 命令的主配置目录，一般系统启动默认要加载的模块放在/etc/modprobe.conf（较早期的系统为 modules.conf）。在主配置目录中，主要存放模块的“黑名单”（blacklist），所谓黑名单，其实是系统为了避免某一个硬件同时有两个以上的模块支持，可能造成模块与模块间的问题，才建立的一个黑名单的机制，让用户先行配置那些不使用模块，并让正确的模块可以正常运行。

◆ netplug 及 netplug.d

这两个目录和网络接口的联机与否有直接关系，因为主要是控制联机时的接口操作。

- netplug：配置有哪些须要控制的网络接口，默认为“eth*”，也就是所有以太网接口，当这些接口有网线连接的时候，就会再参考【netplug.d】目录中所定义的操作。
- netplug.d：定义当网络接口联机时，该做哪些操作，一般的配置直接会启动该网卡，然而，因为这是以 daemon 的方式检测，所以须要使用这个系统功能，需要将“netplugd”服务启动。

◆ opt

此目录原本是定义为存放所有额外安装软件的主机配置文件（也就是该软件和主机相关的配置），但目前并没有被使用到，系统安装时只是先建立这个空目录，软件也没有将此类配置文件放置其中，这可能要等待一段时间，厂商才会开始配合。

◆ pcmcia

这是 PCMCIA 的配置文件目录。当 USB 设备还未盛行时，PCMCIA 是笔记本电脑不可或缺的接口，需要即插即用的方式，即 PCMCIA。但伴随着 USB 接口的出现，以及到目前为止各厂商的支持，现在如果要用户买一台有 PCMCIA 的新笔记本电脑，大概几率很低了！因为 USB 都不够用，还要浪费位置放 PCMCIA 是不可能的事，笔者现在已经有好久没看到这样的设备了，此目录不知还可以存在多久。

◆ pm

由 pm-utils 组件所提供的目录，pm-utils 是一套电源管理的工具软件，在 2.6 的 kernel 之后，都支持 ACPI 的功能，因此可以做到像 Hibernate 的省电模式那样，但这些操作在启动时，若还有硬件（像 USB 设备）在系统上该如何处理，可以通过 pm-utils 来做一些关闭硬件的操作，但由于 pm-utils 是通过一个“HAL”的服务程序（在 Redhat 中此服务的文件名称为 haldaemon）在进

行控制的，所以一般看不到 pm 的字眼。虽然可通过 HAL 操作，但用户同样可以在其中加一些自己的操作，做细节的调整。

基本上，pm-utils 除了【/etc/pm】这一目录外，【/usr/lib/pm-utils】也是主要的目录之一，其实，如果细看【/etc/pm】，里面是空白的，因为这是预留给用户自行使用的地方，系统真正执行的是【/usr/lib/pm-utils】底下的文件，然而，【/etc/pm】是可以使用的，只是文件要自行产生。

在【/usr/lib/pm-utils】下，最好用的目录就属 sleep.d 了（如图 6-70 所示），因为当用户在进行 Hibernate（休眠模式）时，就会依据此目录中文件名前两位数字的大小（由小到大），按序执行操作（正常应该是关闭的操作，但因为可以自行配置，所以不在这里定义为关闭）；相同的，在【/etc/pm】目录下，也有和这里类似的目录，只是都是空的，这在上面有提过。但这里还请特别注意以下几点：

- 系统在进行 Hibernate 时，先执行【/etc/pm】目录，再执行【/usr/lib/pm-utils】目录中的文件；但启动是相反的，会先执行【/usr/lib/pm-utils】，再执行【/etc/pm】目录下的文件。
- 系统在进行 Hibernate 时，会从目录中文件名的前两位数字由小到大执行操作；但再启动时，是相反的，也就是由大到小执行。

这两点要注意的原因，主要是一般默认立场要进行关闭的操作，所以往往会忽略开机时要进行启动，而目的刚好相反，自然顺序也就跟着变动，在细节的设计上要特别注意。

```
[root@LinuxTree sleep.d]# pwd
/usr/lib/pm-utils/sleep.d
[root@LinuxTree sleep.d]# ls
00clear   05led             49bluetooth  60sysfont  94cpufreq
01grub    10NetworkManager  50modules    65alsa     95led
02infome  20video           55battery    90clock    99video
[root@LinuxTree sleep.d]#
```

图 6-70：所有 pm-utils 目前会执行的项目

◆ ppp

PPP（Point-to-Point Protocol；点对点协议）相关的配置文件都放在此目录下，PPP 是各家 ISP（网络服务供货商，像 Hinet 或 Seednet）用来让所有用户联机上网的通信协议之一，位于 OSI 的第二层（数据链接层），相对大家常听到的协议，在同样位置有以太网或无线所使用的 802.11a/b/g 等，都一样是数据链接层的协议，不过现在的 ADSL 已经不再使用 PPP（或是更早以前的 SLIP）了，取而代之的是 PPPoE（PPP over Ethernet），也就是利用以太网的数据包，传送 PPP 的协议，这样就可以直接使用以太网的环境进行拨号，不然之前的 PPP 联机都是通过 Modem

进行的。

◆ profile.d

这一个目录存放的是系统部分的软件配置，但会按不同的 shell 执行不同的文件，默认所使用的 bash，会直接执行该目录中所有扩展名为【.sh】的文件（如图 6-71 所示），其中不乏一些“alias”、“语言”或“软件”的细节配置，但其实 Linux 有类似这样的配置目录还真是非常多（说方便是很方便，但太多的时候，要能掌握关联性就很难）。

```
[root@LinuxTree profile.d]# ls *.sh
ccache.sh   glib2.sh              krb5-devel.sh         less.sh  which-2.sh
colorls.sh  gnome-ssh-askpass.sh  krb5-workstation.sh   qt.sh
cvs.sh      kde.sh                lang.sh               vim.sh
[root@LinuxTree profile.d]# _
```

图 6-71：profile.d 目录下扩展名为 sh 的文件

◆ rc.d

【/etc/rc.d】目录主要用来定义在每一个执行阶段（runlevel）必须要执行哪些系统服务或程序，在目录中主要分为三个重要的部分。

- rc.sysinit：不论用户配置的执行阶段是什么，在系统一开始启动时所遇到的第一个文件，就是该目录下的 rc.sysinit，这一个文件中所记录的，是在服务启动之前所须准备的所有事情，包括启动时会看到的欢迎画面。
- rcX.d 目录：继 rc.sysinit 文件之后所要执行的，就是在【/etc/rc.d】下的 rcX.d 的目录，X 是当初启动配置的 initdefault 值（在/etc/inittab 文件中），若 initdefault 配置为 3，则会转到执行【/etc/rc.d/rc3.d/】下的所有文件。

在目录中最简单的区分方式就是看文件名（如图 6-72 所示），文件名一律都由两个英文字母开头，不是 K 就是 S。K 代表的是 Kill，而 S 代表的是 Start。意思很简单，默认为启动要执行的服务，其文件名的开头就是 S，若用户将某个服务关闭（不论用哪一种工具程序），该文件名就会由 S 开头变为 K 开头。反之亦然，若将某服务关闭，则文件名便会以 K 开头。

当然，这个目录中的文件多少和当初安装操作系统时所选择的项目有关，绝大部分都是服务程序，但是在 rcX.d 目录中的文件，其实都是一些“链接文件”，绝大部分指向【/etc/rc.d/init.d】目录中相对应的文件名称，因为以 link 方式的做法弹性比较大，系统只须要通过这些链接文件知道哪些服务要启动就够了，其中唯独只有一个例外，就是 S99local 这个文件，也就是链接到下面要介绍的【rc.local】文件。

```
[root@LinuxTree rc.d]# ls rc3.d/ -C --color| head -5
K01dnsmasq                  K73winbind          S18rpcidmapd
K01smolt                    K73ypbind           S19rpcgssd
K02avahi-dnsconfd           K74nscd             S22messagebus
K02NetworkManager           K74ntpd             S24openct
K02NetworkManagerDispatcher K76openvpn          S25bluetooth
[root@LinuxTree rc.d]#
```

图 6-72：/etc/rc.d/rc3.d 目录下的部分文件

- rc.local：从刚刚【rc3.d】目录下的文件名就可以知道 S99local 是最后一个要执行的程序，因此，才会将数字部分以“99”来命名，等所有服务执行完后才会轮到它。该文件链接到的是“/etc/rc.d/rc.local”文件，若把该文件打开会发现什么都没有，里面只有一行，记录要产生指到/var/lock/subsys/local 文件（其目录中存放的是代表正在执行的 daemon 文件，都是空文件，只是为了记录有哪一个服务被毫无警告地突然中止）。但这个文件并不是每一个 Linux Distribution 都有的，像 SuSE 就不叫做 rc.local，而是使用在/etc/rc.d/下的 boot.local 文件来做配置。因此，每一套操作系统都应该会有自定的个人化配置文件，只是必须由用户自己找出来，如图 6-73 所示。

```
[root@localhost rc.d]# cat rc.local
#!/bin/sh
#
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local
[root@localhost rc.d]#
```

图 6-73：/etc/rc.d/rc.local 文件的默认内容

◆ readahead.d

这个目录是“readahead”组件主配置目录，“readahead”是一个聪明的机制，大家知道目前计算机的执行速度大部分都是被硬盘的读取速度所拖慢，因此，为了加速操作系统的使用速度，readahead_early 及 readahead_later 这两个 daemon 在系统加载时，直接将日常所需要的一些文件，全部先放到硬盘的高速缓存（cache）中，让文件直接通过 chche 读取。

然而，readahead.early 及 readahead.later 的差异，在于一个先启动，一个后启动（这从文件名就可以看出来，好像有点废话），若用户可以细看这个目录下的配置文件（扩展名为 early 及 later），会发现最主要的差异在于 readahead.early 所要预先加载的是一些基本系统所需的文件，而 readahead.later 则是一些 X Windows 所需的文件；另外，从/etc/rc.d/下的 runlevel 3 与 runlevel 5 的目录中（如图 6-74 所示），也可以发现在 runlevel 3 时，readahead_later 是被关闭的，但在 runlevel 5 时，则是启动的，而且被排在很后面的阶段（因为其文件名最前面的数字为 96）。

因此，我们大概可以知道，readahead 的服务，分为 early 与 later 两个阶段，在/etc/readahead.d 目录中自然就有两个针对这两个阶段的配置文件，而 early 负责一般系统的文件，later 则负责以 X Window 为主的文件。

```
[root@LinuxTree readahead.d]# ls /etc/rc.d/rc3.d/*read*
/etc/rc.d/rc3.d/K99readahead_later  /etc/rc.d/rc3.d/S04readahead_early
[root@LinuxTree readahead.d]# ls /etc/rc.d/rc5.d/*read*
/etc/rc.d/rc5.d/S04readahead_early  /etc/rc.d/rc5.d/S96readahead_later
[root@LinuxTree readahead.d]#
```

图 6-74：readahead 服务在 runlevel 3 和 5 之间的差别

◆ redhat-lsb

这和上面提到的【lsb-release.d】目录都是由同一个组件“redhat-lsb”所提供的相似（何谓 LSB 请参考 lsb-release.d 目录的说明），但在这个目录中，已经有提供现成 Redhat 按照 LSB 所开发出的一些软件，也就是说，只要执行这个目录下的程序（如图 6-75 所示）即可，假设为【lsb_pidofproc】，结果应该会和所有 Linux Distribution 中符合 LSB 的【lsb_pidofproc】程序一样。但从图 6-75 可以发现，目前 Redhat 的 LSB 作法，都是将其功能放在 functions 的文件里面，因此，基本上一定要通过 functions 才可以使用 LSB 的格式。

```
[root@LinuxTree redhat-lsb]# ls
lsb_killproc  lsb_log_message  lsb_pidofproc  lsb_start_daemon
[root@LinuxTree redhat-lsb]# cat lsb_pidofproc
#!/bin/bash

. /etc/init.d/functions

pidofproc $*
exit $?
[root@LinuxTree redhat-lsb]#
```

图 6-75：/etc/redhat-lsb 目录下的文件及示例

◆ rwtab.d

这个目录是一个在启动时会去参考的目录，主要的文件在【/etc/rwtab】，在【/etc/rwtab.d】目录中的目录或文件都是准备和【/etc/rwtab】一起被系统所加载的部分。在其中任何一个文件中，分为两个字段 type 和 path，而 type 又可分为 empty、dirs 及 files 三种。

通过这些文件内容，【/etc/rc.d/rc.sysinit】这一负责加载系统的文件，会将里面所有列出的文件及目录先暂存放到另一个区域中，以免系统出现问题时文件全部被清掉，算是一个系统初期的备份机制。

◆ sane.d

这是在系统下要使用扫描仪所需的配置目录，它的主要配置文件为【sane.conf】，sane（Scanner Access Now Easy）为了方便用户在各式的扫描仪连接时都可以使用，因此，在这一目录中放置了很多种不同类型扫描仪的硬件信息（文件名也都以型号命名），让系统在检测到扫描仪时可以直接使用。而在实际的使用上，sane 组件（sane-backends）提供两个程序供用户使用，一个是【sane-find-scanner】，用来检测主机上所连接的扫描仪；另一个是【saned】daemon，负责让远程的用户使用本地的扫描仪。

◆ setuptool.d

这个目录是【setuptool】系统配置组件的主要配置目录，对经常须要使用【setup】命令的用户来说非常方便，因为可以完全自定义【setup】命令中所有的配置项目及执行顺序。

项目部分，只要在此目录中增加一个文件即可（如图 6-76 所示，增加一个 98kernel 的文件），只要在文件中注明“执行文件位置 | 类别”这两个字段即可，当执行 setup 命令时就可以看到。

```
[root@LinuxTree setuptool.d]# ls
98kernel                          99kbdconfig       99system-config-network-tui
98netconfig                       99lokkit          99system-config-printer-tui
98system-config-authentication     99mouseconfig     99timeconfig
98system-config-display            99ntsysv          99Xconfigurator
98system-config-keyboard           99printconf-tui
99authconfig                      99sndconfig
[root@LinuxTree setuptool.d]# cat 98kernel
/root/makekernel|Compile configuration
[root@LinuxTree setuptool.d]#
```

图 6-76：在/etc/setuptool.d 目录中增加选项

顺序部分，以文件名前面两位为主，越小的越前面执行，如图 6-76 中的【netconfig】与【system-config-network-tui】两个文件，分类都在“Network Configuration”，但默认为先以【netconfig】命令操作，若该命令出现问题（或者是没有安装该组件），就会执行【system-config-network-tui】命令。

笔者在这个目录中新增一个【98kernel】文件及内容后，再执行【setup】命令（见图 6-77），就可以看到“Compile Configuration”的选项，而这也就是刚刚在这个文件中属于类别的部分，是一种相当方便的配置方式。

◆ skel

系统管理员大部分都很喜欢这一目录，尤其当系统中用户数量很大，以及须要让每个用户在主目录中有些特定文件时，因为这一目录就是用户主目录一开始所需的文件，当系统在建立一个

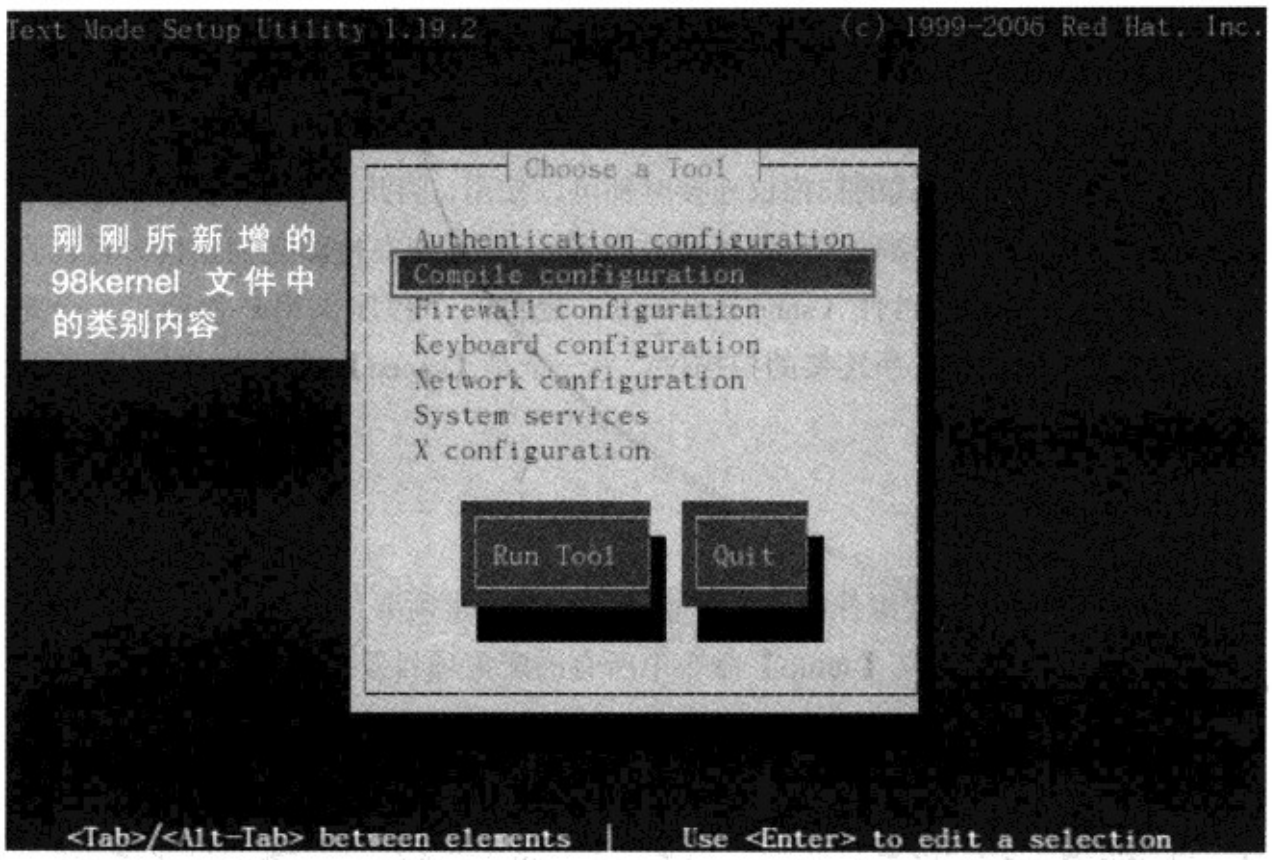


图 6-77：变更过后的 setup 命令选项

新用户的时候，会从【/etc/skel】目录中把所有文件复制一份到【/home/newuser】的路径下，换句话说，当用户刚被建立时，其主目录中的文件和【/etc/skel】下的文件是一模一样的（如图 6-78 所示）。也因为如此，当管理员需要让每一个用户都拥有某些非系统默认的目录或文件时（如每一个用户的网页目录或文件），就可以先将这些文件建立在这目录中，这样即便是临时新增的用户，也无须担心会少掉某些文件或是和其他用户的数据不同。

```
[root@LinuxTree skel]# ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .kde  .xemacs  .zshrc
[root@LinuxTree skel]# ls -a /home/newuser/
.  ..  .bash_logout  .bash_profile  .bashrc  .emacs  .kde  .xemacs  .zshrc
[root@LinuxTree skel]#
```

图 6-78：/etc/skel 目录与新用户主目录中文件之比对

◆ sysconfig

这是一个非常重要的系统配置文件的存放目录，里面放置了大量系统启动及运行相关的配置文件，从系统启动时（initrd 刚结束之后）的配置，一直到进入系统后的主机配置，大部分都可以在这个目录中找到（如图 6-79 所示），图 6-79 中的 clock 就是系统时区的配置，而 i18n 则是系统的语言，这两个只是上百个配置文件中的两个，另外如防火墙软件（iptables）的配置也在这里。

```
[root@LinuxTree sysconfig]# pwd
/etc/sysconfig
[root@LinuxTree sysconfig]# grep -v ^# clock
ZONE="Asia/Taipei"
UTC=false
ARC=false
[root@LinuxTree sysconfig]# grep -v ^# i18n
LANG="en_US.UTF-8"
SYSFONT="latarcyrheb-sun16"
[root@LinuxTree sysconfig]#
```

图 6-79：在 sysconfig 目录下的两个示例文件

因此，这里面的文件多不胜数，只要可以想到的系统配置，先到这个目录中查找就可以，除了目录下的文件外，也有一些重要的子目录在 sysconfig 的目录下，最常用的如【network-scripts】，也就是网络接口的 IP 地址记录的地方，实际还有非常非常多的配置，如一些服务、软件、硬件或主机相关配置等，全都在这个目录中。

如果要了解到底有哪些是被执行的，可以参考【/etc/rc.d/rc.sysinit】中的内容，因为大部分所执行的都是从这个 script file 开始的，只是其中的内容有点复杂，或者可以参考笔者的著作《Linux 操作系统之奥秘》第 5 章的内容，我们已经整理成一份表格，会比较清楚。

◆ syslog-ng

syslog-ng 为新一代的系统记录服务（syslog Next Generation），既然是新一代，当然比原本的 syslog 要强得多，除了一般系统与服务的记录能力之外，还可以做到以下几点。

- 支持通过 TCP/UDP 的协议，收集整个网络的相关 log 信息，以及加密的技术，提高整体的安全性。
- 可以用正规表达式，将系统记录的信息自行定义出个别的分类及处理方式，以前的 syslog 是完全的照单全收。
- 可以将所收集到的 log，利用管道（pipeline）或重定向（redirect）方式，交由其他程序处理后再存成记录，这样可以让管理员对系统的 log 信息，实时地做更深层次的处理（但到的 log 文件就更有意义）。

但由于这一套程序不完全是 Open Source，有一些高级的功能须要购买，所以像 Redhat 默认并没有采用这一套 syslog-ng 的 log 管理程序，因此在这个目录下仍是空的。

◆ udev

udev 是一套设备的管理机制，udev 通过 sysfs 的文件系统（也就是要参考/sys 目录下的硬件信息），可以正确地掌握目前系统上存在的硬件设备，以及针对每一个硬件设备做出不同的判断

与执行。这个目录的重要性，在于可以决定 udev 要针对哪些硬件做出哪些操作，换句话说，可以让管理员在启动时、系统执行中、硬件插入或拔出时做出实时的操作（包括改硬件名称），以反映目前系统的需求，这是很棒的一件事。

在【/etc/udev】目录中有一个【rules.d】的子目录，里面所存放的是目前系统针对不同的硬件设备的操作，有很多用户每天都在用但不知道是 udev 所提供给用户的规定，例如：光驱。

以前在使用光驱之前，必须先确定光驱是 IDE、SCSI 或 USB 接口，再来针对不同的接口，检查光驱本身该在接口下的哪一个位置，这样才能知道光驱的设备文件是【/dev/hdb1】或【/dev/sdb1】（只是示例，实际以使用时为主），但现在大家都不用管这件事，只要知道当使用光驱时，直接将【/dev/cdrom】挂载到其他目录，就可以使用，这件事是由 udev 的机制所提供的。

从图 6-80 中可以看到，在【/etc/udev/rules.d】中有许许多多的规则文件，文件名的前两位为执行的顺序（也就是硬件规则的优先级），每一个规则文件中都保存有很多的判断方式与结果。以刚刚提到的光驱为例，在【50-udev.rules】中（这个文件其实是 udev 很重要的一个规则文件，因为/dev 下的设备文件是从这里生出来的），有一段内容是有关光驱的设备文件产生的规则，这段文字的意思就是“不论是哪一种接口的光驱，全部建立对应的‘cdrom’及‘cdrom-接口名称’的链接文件”，所以当系统上有光驱时，用户可以看到【/dev/cdrom】文件，当光驱移除掉，这个文件就不见了（不需要重启，是实时的），针对 udev 更详细的说明，请参考笔者著作《Linux 操作系统之奥秘》的第 4 章。

```
[root@LinuxTree rules.d]# ls
05-udev-early.rules  60-libsane.rules      90-alsa.rules
10-libifp.rules      60-net.rules          90-hal.rules
40-multipath.rules   60-openct.rules       95-pam-console.rules
50-udev.rules        60-pcmcia.rules       99-fuse.rules
51-udr.rules         60-wacom.rules        bluetooth.rules
60-libmtp.rules      85-pcscd_ccid.rules   xen-backend.rules
60-libnjb.rules      85-pcscd_egate.rules
[root@LinuxTree rules.d]# grep cdrom 50-udev.rules
KERNEL=="sr[0-9]*",              SYMLINK+="cdrom cdrom-%k"
KERNEL=="scd[0-9]*",             SYMLINK+="cdrom cdrom-%k"
KERNEL=="pcd[0-9]*",             SYMLINK+="cdrom cdrom-%k"
KERNEL=="hd[a-z]", BUS=="ide", ATTRS{removable}=="1", ATTRS{media}=="cdrom", SYM
LINK+="cdrom cdrom-%k"
```

图 6-80：/etc/udev/rules.d 下所有的规则文件及 cdrom 相关的部分内容

◆ xen

Xen 是 Linux 下目前虚拟化很有名的一种解决方案，通过 Xen 的机制，可以在 Redhat 上安装 Redhat 本身、SuSE、Solaris、Windows 等各种操作系统，它大致可以分为两种虚拟方式：PV 和 FV。

PV（Para-Vritualization）和 FV（Full Virtualization）的差别，主要以 guest OS 的硬件模拟程度做区分，详细说明如下：

- FV：FV 是一般较常看到的做法，所有的 guest OS（也就是虚拟出来的操作系统）完全不会看到实际的硬件是什么，只能使用由 Supervisor 所提供的所有虚拟硬件，因此，在这种机制下，guest OS 运行的性能一定会大受虚拟接口的影响。另外还有一个缺点，就是因为完全模拟的关系，不支持新的技术，连 ACPI（Advanced Configuration and Power Interface）开关机的机制都无法使用，也就是当用户在 FV 的 guest OS 下，若直接触动关机按钮（这里的按钮是 VMM 所提供的，不是主机上的），会直接断电，而不会进行关机程序。
- PV：至于 PV 的做法，有鉴于一般 Virtual Machine 工具都是以完全模拟的方式（也就是刚刚提及的 FV 模式），造成性能上的降低，因此，XEN 在设计上，希望各操作系统可以在开发时就将 XEN 的技术包进去，这样在使用时，就可以用局部模拟的方式，让虚拟操作系统可以直接使用到硬件中的 CPU、内存等（如图 6-81 所示），而无须再经由 XEN 做模拟的操作。

```
[root@LinuxTree xen]# grep -v ^# xmexample1|grep ^ |grep [a-z,1-9]
kernel = "/boot/vmlinuz-2.6.10-xenU"
memory = 64
name = "ExampleDomain"
vif = [ '' ]
disk = [ 'phy:hda1,hda1,w' ]
root = "/dev/hda1 ro"
extra = "4"
[root@LinuxTree xen]#
```

一些虚拟操作系统中硬件设定的范例

图 6-81：/etc/xen 下面某一个 guest OS 的示例文件

而这个目录是 XEN 主要的配置文件所在，不只是 XEN 的配置文件，连 XEN 下面所有的虚拟操作系统的配置文件都在内，一般会使用到的也是虚拟操作系统的配置文件，因为里面可以配置每一个虚拟操作系统的 CPU、Memory、硬盘等基本的硬件配置，更重要的是，整个 XEN 的网络环境，也必须要通过这个目录下的网络配置文件才可以建立出来（这就是用户在系统中用【ifconfig -a】会看到许多 xenbr0、virif0:0、pth0 等虚拟接口的原因），对 XEN 来说，这个目录可以控制虚拟操作系统的所有硬件选项，自然是非常重要的。有关虚拟操作系统的详细做法，请参考《Linux 操作系统之奥秘》的第 9 章，里面有较完整的介绍，包括如何在 Linux 下安装 Windows。

◆ yum 及 yum.repos.d

这两个都是 yum（Yellowdog Updater Modified）的目录，是一套在 Linux 下可以自动帮助用户安装、更新、移除等的组件管理员，未来应该会取代掉目前常使用的 rpm 方式，因为在 yum

的管理下会更为简便。主配置文件在【/etc/yum.conf】是 yum 组件的配置，至于【yum】及【yum.repos.d】，则分别代表【更新方式及外挂程序的配置目录】和【存放定期更新组件内容】的信息，如在 X Window 中有时会出现更新失败或过期的信息，就是因为在【/etc/yum.repos.d】目录中有“update”的文件存在，所以系统就会定期地去做更新操作，当网络不通或无法联机时，就会出现错误，如图 6-82 所示。

```
[root@LinuxTree yum.repos.d]# ls
fedora-development.repo  fedora-updates.repo          pidgin.repo
fedora.repo              fedora-updates-testing.repo
[root@LinuxTree yum.repos.d]#
```

图 6-82：/etc/yum.repos.d 目录中的文件

6.1.4 安全性目录

在【/etc】下和 Linux 操作系统安全性相关的子目录，都可以在这里找到。

◆ audit

这个目录所代表的是一种和目录名称一致的“audit”安全机制，主要以服务（daemon）的方式协助管理员持续监控各文件被存取的情况。这个机制是非常重要的，因为 Linux 中的文件太多太多，但重要的文件一定要保护到，为了系统的安全性、完整性或避免黑客入侵后的某些修改文件操作（被更动的结果很有可能是门户大开），针对某些文件做适度的监控是绝对必要的，而 audit 机制，可以提供给用户非常完整的监控及回报方式。

以 su 命令为例，为了知道有谁曾经切换成管理员的权限，可以利用 audit 的机制来监控该命令的一举一动，在本示例中（如图 6-83 所示），因为 su 一般都是直接执行，所以只以“执行”的权限为监控目标。

Ⓐ 先确认 audit 的服务是否已启动。

Ⓑ 使用 auditctl 命令，制定属于符合需求的监控环境（这有点像 iptables 的架设方式），【-w /bin/su】的意思是监控（watch）“/bin/su”命令；而【-p x】的意思则是只要监控有关“执行”的部分（x 就是文件权限中读（r）写（w）执行（x）中的 x）。

Ⓒ 当没有人使用 su 命令时，使用【ausearch –f /bin/su】命令是不会有任何记录的（no matches），但当有用户要切换为 root 管理员（这里以 juergen 为例，其 uid 为 501），执行过【su -】后，就会发现其记录多出一条有关【su】命令执行的细节，其中执行的命令及谁执行的一定都会记录在其中，其信息很详细，所以很适合当作追查用的信息。

```
[root@LinuxTree audit]# /etc/rc.d/init.d/auditd status
auditd (pid 1727) is running...
[root@LinuxTree audit]# auditctl -w /bin/su -p x
[root@LinuxTree audit]# ausearch -f /bin/su
<no matches>
[root@LinuxTree audit]# ausearch -f /bin/su
----
time->Mon Mar  3 05:47:35 2008
type=PATH msg=audit(1204494455.523:471): item=1 name=(null) inode=2105178 dev=fd
:00 mode=0100755 ouid=0 ogid=0 rdev=00:00
type=PATH msg=audit(1204494455.523:471): item=0 name="/bin/su" inode=1698959 dev
=fd:00 mode=0104755 ouid=0 ogid=0 rdev=00:00
type=CWD msg=audit(1204494455.523:471):  cwd="/home/juergen"
type=EXECVE msg=audit(1204494455.523:471): a0="su" a1="-"
type=SYSCALL msg=audit(1204494455.523:471): arch=40000003 syscall=11 success=yes
 exit=0 a0=83ee498 a1=83decf8 a2=83debc8 a3=0 items=2 ppid=24323 pid=24367 auid=
501 uid=501 gid=501 euid=0 suid=0 fsuid=0 egid=501 sgid=501 fsgid=501 tty=tty1 c
omm="su" exe="/bin/su" key=(null)
[root@LinuxTree audit]#
```

A　B　C

图 6-83：audit 的实务示例

在/etc/audit 目录中，有一个 audit.rules 文件，若用户希望每次启动时，都会自动监控某些文件，就可以将欲监控的方式，先写在这个文件中，这样，每次启动后都会由 audit 进行监控。

◆ pam.d

此目录是 Linux-PAM（Pluggable Authentication Modules for Linux）的所有配置文件，配合【/lib/security】目录中所有的函数库，提供 Linux 下的应用程序认证的机制。

每一个 PAM 的规则都是以下面五个区块做控制（如图 6-84 所示，包括文件名也都是一种控制条件）的：

- service。
- type。
- control。
- module-path。
- module-arguments。

```
[root@LinuxTree pam.d]# head -3 vsftpd
#%PAM-1.0
session    optional     pam_keyinit.so    force revoke
auth       required     pam_listfile.so   item=user sense=deny file=/etc/vsftpd/ft
```

service　A　B　type　C　control　D　module-path　E　module-arguments

图 6-84：PAM 配置规则的基本架构

在此分别将五个区块详述如下：

Ⓐ service：在 Redhat 目前的 PAM 实现上（也可以直接写在/etc/pam.conf 中），都是以【/etc/pam.d】目录中的“文件名”为 service 的值，所以一定都要以小写代表，不然是无法使用的。

Ⓑ type：PAM 所使用的认证模块分为四类（type）：

- account：在登录后可针对每一个用户做用户属性（如 uid、gid、本地或远程用户等）的细节检查。
- auth：用来做身分识别的操作，也就是一开始的登录模块。
- password：改变授权用户的方式，最常看到的就是专门针对密码在修改时的一些限制，包括密码字典（就是普通用户用简单密码会无法通过的验证方式）。
- session：一般是用户在登录前与注销后（正确地说是取得该项服务的前后时间点）的操作。

Ⓒ control：在认证过程中，分为五种控制方式（也就是当成功或失败时的处理方式）：

- required：这一种方式不论该次模块的成功或失败，都会继续下一个步骤，直到全部完成后（同一堆栈模块中的 service 和 type），才会回传失败的结果。
- requisite：和 required 类似，但当该模块失败时，就会直接回传失败的结果。
- sufficient：在同一堆栈的模块中，只要该模块成功，就不会再往下继续。但当前面有 required 的方式，并为失败的结果，该 sufficient 的模块即使成功，也会被忽略（也就是会继续往下运行）。
- optional：这一种方式只对在同一组合【service+type】中唯一的 optional 模块有用，而且不会影响其他模块的运行。
- include：将所指定的配置文件中所有和该模块一样 type 的规则全部加入。

Ⓓ module-path：在该次认证中所须使用到的模块名称，这些模块文件可以在【/lib/security】下面找到。

Ⓔ module-arguments：针对该模块可以接受的参数。

图 6-85 是一个简单的示例，要将所有用户的登录机制变宽松（不论输入的密码正确或错误，都允许用户登录）。

Step 1 login 服务是主要的配置文件，但其登录的认证交给 system-auth 服务处理。

Step 2 修改 system-auth 服务的内容，将白色区域的模块先用“#”mark 起来，让其失效。

Step 3 几秒钟后，就可以发现用户名称只要正确，密码随便输入，都可以登录 Linux。

这是因为在 auth 部分（模块的堆栈），用蓝色所选中的部分，该模块是用来做密码验证的，但不管结果如何，都会进入下一步（因为其方式为 sufficient，当然读者可以将其改成 required，就一样不可以登录了），而下面同 type 的模块都被 mark 掉，所以自然认证部分就结束，也因为该 type 的堆栈中最后的是 sufficient 方式，因此顺利通过。

```
[root@LinuxTree pam.d]# head -5 login
#%PAM-1.0
auth [user_unknown=ignore success=ok ignore=ignore default=bad] pam_securetty.so
auth       include      system-auth
account    required     pam_nologin.so
account    include      system-auth
[root@LinuxTree pam.d]# head -8 system-auth
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
auth        requisite     pam_succeed_if.so uid >= 500 quiet
auth        required      pam_deny.so

[root@LinuxTree pam.d]# head -8 system-auth
#%PAM-1.0
# This file is auto-generated.
# User changes will be destroyed the next time authconfig is run.
auth        required      pam_env.so
auth        sufficient    pam_unix.so nullok try_first_pass
#auth        requisite     pam_succeed_if.so uid >= 500 quiet
#auth        required      pam_deny.so

[root@LinuxTree pam.d]#
```

图 6-85：登录程序与 PAM 机制的部分关系

◆ pam_pkcs11

这只是上述 PAM 机制中的一种登录模块（PAM PKCS#11），可以让用户通过 smart card 做登录的操作，但一般系统上，目前默认在 PAM 的配置中没有使用这一项，有兴趣的读者请参考其网站介绍：

http://www.opensc-project.org/pam_pkcs11/

◆ pki

PKI（Public Key Infrastructure）是一种公开密钥（加密验证的一种模式）的管理方式，通过这样的管理模式，可以让所有网络传输有更多保障。简单地说，PKI 的架构下（如图 6-86 所示）分所谓的公钥（Public Key）与私钥（Private Key），公钥是任何人都可以拿到的，而私钥只有自

己才有。

而当我们在网络上要传输数据时，可以利用对方的公钥将文件加密，送给对方，对方就用自己的私钥解密，这个过程就是 PKI 的基本精神。有些人可能会觉得很难懂，但基本上其实每天都在用，也就是我们的浏览器（像 IE），当我们在浏览网站时，会有很多的“凭证”出现，需要我们同意，这些凭证就是“公钥”，而我们同意时就拿“私钥”解开，当下次出现凭证相关画面时，就要知道这就是 PKI 在做的事情。

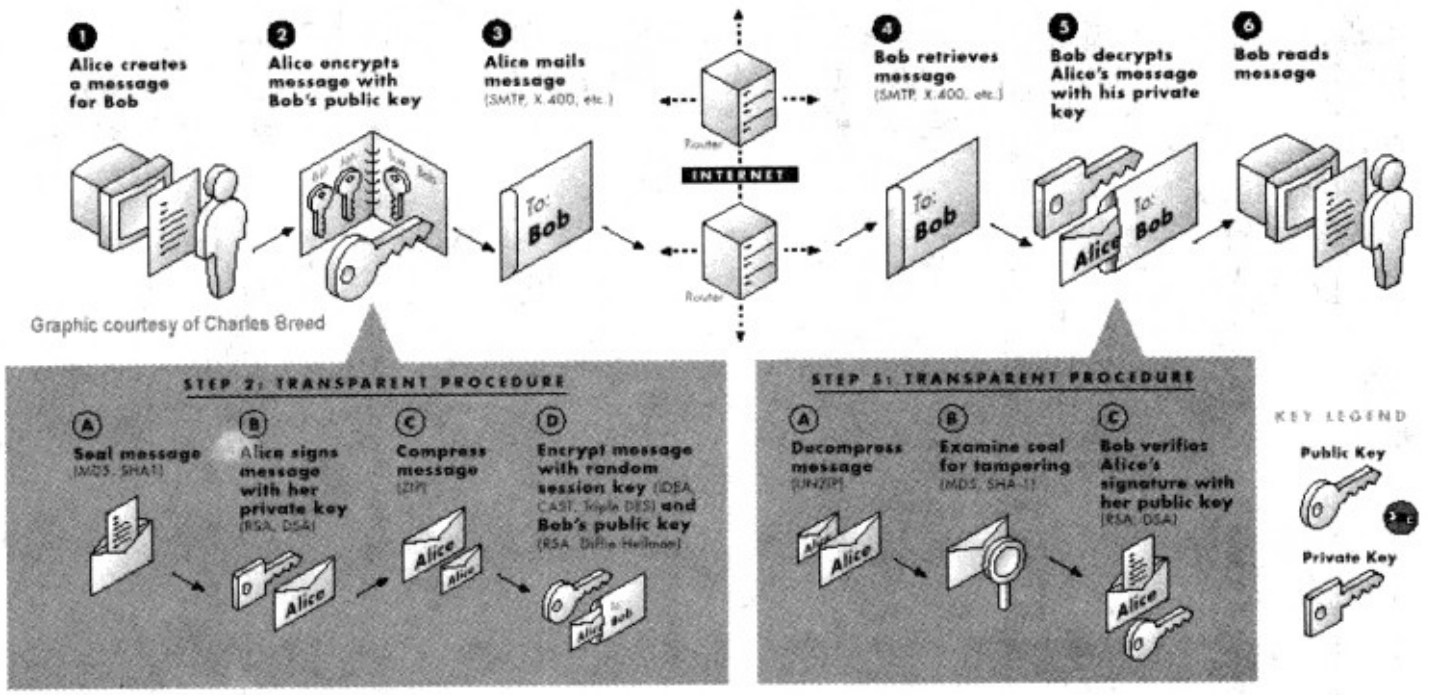


图 6-86：PKI 的架构图[2]

基本上，Linux 只是建立一个空目录，有须要用到的软件就自行将其认证信息存在 pki 的目录中（如图 6-87 所示），包括 Redhat 本身 RPM 网络安装所需的认证文件也在【rpm-gpg】的目录中。

```
[root@LinuxTree pki]# pwd
/etc/pki
[root@LinuxTree pki]# rpm -qf /etc/pki
filesystem-2.4.6-1.fc7
[root@LinuxTree pki]# rpm -qf *
bittorrent-4.4.0-5.fc7
openssl-0.9.8b-12.fc7
dovecot-1.0.0-11.fc7
nss-3.11.5-2.fc7
fedora-release-7-3
openssl-0.9.8b-12.fc7
[root@LinuxTree pki]#
```

其中的文件都是由各软件自行安装的

图 6-87：/etc/pki 目录下的文件来源

[2] 图片来源：*http://www.smartcardbasics.com/security_3.html*。

◆ racoon

这个目录是由 ipsec-tools 组件所提供的，IPsec 的主要目的是在让系统实现 VPN 的网络技术，因为在传输之间会有密钥（Key）交换的管理问题，所以，这一目录的名称 racoon 所代表的，就是 Linux 在 IPsec 技术中，主要是管理密钥的协议（IKE，ISAKMP/Oakley）及服务名称，在 racoon 目录的主要配置文件【racoon.conf】中，定义在 IPsec 实作上所需要的加密算法种类（如 des、3des 及 aes 等）及其他细节的配置，在【racoon】及【racoon.conf】的 man page 中有详细的介绍及配置方式。

◆ security

这个目录和前面提到的【pam.d】目录是相辅相成的，因为在【pam.d】中所有 PAM 的规则，都要用到【/lib/security】下的 PAM 函数库，但光看这些函数库的使用方式，很难理出一些头绪（但用户是可以针对每一个函数库去看其 man page 的，如【man pam_limits】），在【/etc/security】目录中，就是针对这些函数库，提供以配置文件的方式进行细节配置，对希望调整系统安全性部分增加了非常大的方便性，如图 6-88 所示。

```
[root@LinuxTree security]# ls /lib/security/
pam_access.so          pam_issue.so          pam_passwdqc.so       pam_tally.so
pam_ccreds.so          pam_keyinit.so        pam_permit.so         pam_time.so
pam_chroot.so                                pam_pkcs11.so         pam_timestamp.so
pam_ck_connector.so    pam_krb5afs.so        pam_postgresok.so     pam_umask.so
pam_console.so         pam_krb5.so           pam_rhosts_auth.so    pam_unix_acct.so
pam_cracklib.so        pam_lastlog.so        pam_rhosts.so         pam_unix_auth.so
pam_debug.so           pam_ldap.so           pam_rootok.so         pam_unix_passwd.so
pam_deny.so            pam_limits.so         pam_rps.so            pam_unix_session.so
pam_echo.so            pam_listfile.so       pam_securetty.so      pam_unix.so
pam_env.so             pam_localuser.so      pam_selinux.so        pam_userdb.so
pam_exec.so            pam_loginuid.so       pam_shells.so         pam_warn.so
pam_faildelay.so       pam_mail.so           pam_smb_auth.so       pam_wheel.so
                       pam_mkhomedir.so      pam_smbpass.so        pam_winbind.so
pam_filter.so          pam_motd.so           pam_stress.so         pam_xauth.so
pam_ftp.so             pam_namespace.so      pam_succeed_if.so
pam_group.so           pam_nologin.so        pam_tally2.so
[root@LinuxTree security]# pwd
/etc/security
[root@LinuxTree security]# ls
access.conf        console.perms     namespace.conf   pam_winbind.conf
chroot.conf                          namespace.init   time.conf
                   group.conf        opasswd
console.handlers   limits.conf       pam_env.conf
[root@LinuxTree security]#
```

对应到 /lib/security 目录中的文件

图 6-88：/lib/security 和/etc/security 文件的比对

在【/etc/security】下的文件，都是【/etc/security】目录中函数库的配置文件，所以有这三个目录（/etc/pam.d、/etc/security、/lib/security），就可以通过【/etc/pam.d】先配置基本的筛选工具

及条件，再通过【/lib/security】下的工具，去读取【/etc/security】下的细节配置，在这样一系列的 PAM 机制，让整个安全性的设计弹性大大提高，对管理员也方便许多。

◆ SELlinux

SELinux（Security-Enhanced Linux）是由美国国家安全局（NSA）所推出的一个很新的安全性方案，它是一种针对各种文件、目录、设备或 daemon 等在 Linux 所须使用到的安全性机制，而且其安全性的数据是直接记录在文件系统中的。目前在 X Window 下虽然有可以配置的接口，但仍不够完整，所有细节的配置还必须在【/etc/selinux】目录下操作，其主配置文件为此目录下的【config】，最基本的 SELinux 开关就在这个文件中。因为笔者对此安全性的机制尚未有时间细看（看起来颇复杂），因此就不误导读者，但如果有兴趣的读者，目前可能只能先参考下方 NSA 官方网页上的信息（如果只是大略调整或开启关闭之类的，就直接用 X Window 下的配置工具即可）。

http://www.nsa.gov/selinux/

◆ wpa_supplicant

这个目录被归类在安全性目录中，是因为属于无线中安全认证的部分，它只有一个【wpa_supplicant.conf】主要的配置文件，用户可以在这个目录中加入已知可登录的 AP（也就是手上有该 AP 账号密码的区域，像常接入的公司或家中的 AP），但也因为这个文件中势必会存放认证所需的密码相关信息，因此这个主配置文件的安全性也十分重要。至于这个配置文件的使用方式，结构不会太复杂，用户可以直接参考 wpa_supplicant 组件所提供的示例文件，wpa_supplicant 将该示例文件默认存放在【/usr/share/doc/wpa_supplicant-0.5.7/wpa_supplicant. conf】。

6.1.5 X Window 目录

在【/etc】目录下，有许多一般用户没想到和 X Window 相关的配置，笔者整理在这个章节中，让读者可以尽量调整自己的 X Window 配置。

◆ alternatives

在 Windows 下有一个可辨识扩展名的“文件类型”选项，可针对同一类型的文件，选出一个默认用户所要使用的程序去执行。如【.txt】文件，可以打开 txt 格式的最少有 Word、Wordpad 及 Notepad（记事本），但默认以记事本去开启 txt 的文件格式，若用户觉得记事本的功能太少，就可以换成另一个程序，如 Wordpad。

在 Linux 一样有这样的功能，就是 Alternatives 组件所提供的，在/etc/alternatives 目录下有所

有目前已经定义的程序名称，都以软链接（Symbolic Link）的方式存在，里面每一个文件其实都有定义好的默认执行程序，可以用【alternatives】的命令查看及修改其配置（如图 6-89 所示）。

```
[root@LinuxTree alternatives]# pwd
/etc/alternatives
[root@LinuxTree alternatives]# ls -l mta
lrwxrwxrwx 1 root root 27 2008-03-03 02:44 mta -> /usr/sbin/sendmail.sendmail
[root@LinuxTree alternatives]# alternatives --config mta

There are 4 programs which provide 'mta'.

  Selection    Command
-----------------------------------------------
   1           /usr/bin/esmtp
** 2           /usr/sbin/sendmail.sendmail
   3           /usr/sbin/sendmail.postfix
   4           /usr/sbin/sendmail.exim

Enter to keep the current selection[+], or type selection number:
[root@LinuxTree alternatives]#
```

有+号代表为默认的执行程序

图 6-89：alternatives 所定义的默认执行程序

这些配置做好之后，在 X Window 下就只要配置默认的执行程序，指定到 alternatives 所配置的文件（如图 6-89 中的 mta），往后只要知道执行 mta，不论实际的执行程序是哪一个，一定都可以连接到正确的程序。

简易的配置可以参考以下的网址：*http://www.linuxquestions.org/linux/answers/Applications_GUI_Multimedia/LINUX_ALTERNATIVES_HOWTO*

◆ fonts

X Window 下大家最重视的一个问题就是“字体”，其中的一个解决方案就是采用 Fontconfig 的组件，而【/etc/fonts】目录就是 Fontconfig 组件的最主要配置文件的存放区。但要注意的是，Fontconfig 所提供的只是一个字库的机制，并不能算是一个软件，也就是说，Fontconfig 主要目的是希望所有在 X Window 上的软件，都可以共享这一个机制，更简单地说，所有软件字体的主要配置文件只需要一个【/etc/fonts/fonts.conf】就够了，这样可以大量简化在 X Window 下字体的复杂度，但当然仍需各软件的配合。

【/etc/fonts】目录下的配置都是以 XML 的方式配置（如图 6-90 所示）的，除了 fontconfig 的主要配置（如图 6-90 中默认搜寻字体的路径）外，还有很多各种字体的默认配置文件也都在其中，对每一种语言也都各自有其独立的目录。

图 6-90：Fontconfig 的主要配置文件中的路径配置

◆ gconf

这一目录是由 GConf2 的组件所建立的，GConf 的作用就是提供 GNOME 下的应用程序注册的机制，让每一个应用程序都有其专属的键值（Key/Value），这和 Windows 下的 regedit 所呈现的软件注册信息是类似的。而 gconf 的目录就是为了存放本身的参考路径之默认值（如图 6-91 所示）与 GNOME 下相关软件其键值的目录。参考路径的种类在其分类下可分为三种：

Ⓐ 强制性配置：在该路径下的所有键值的属性，都是只读的，所以用户无法修改里面的信息。

Ⓑ 用户自定义：在主目录下的键值目录，会放在主目录下，目的就是为了让所有用户可以按自己需求做修改。

Ⓒ 默认值配置：系统所默认的键值，也只可接受读取。

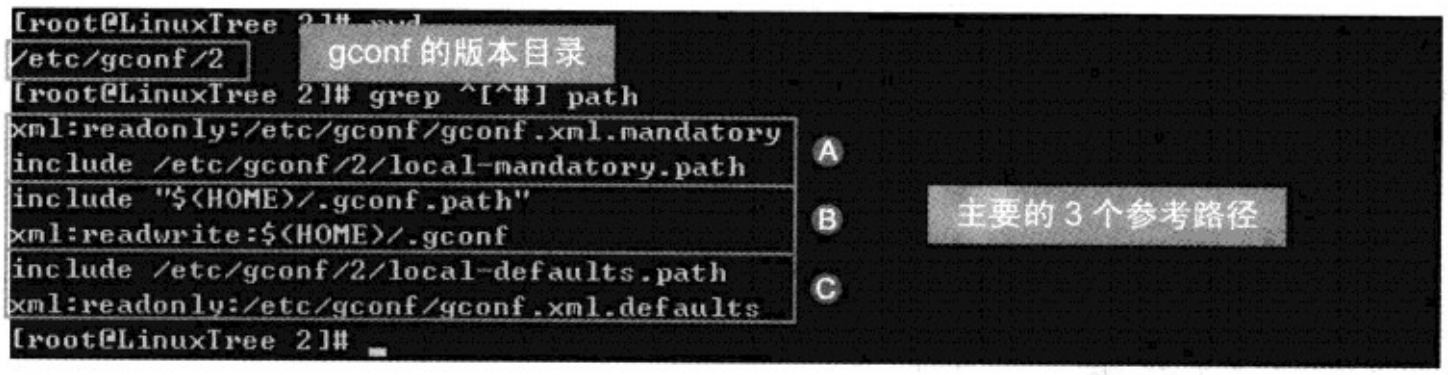


图 6-91：gconf 默认会读取配置文件的路径

而在/etc/gconf 目录中的【schemas】子目录所存放的文件，除了目录本身外，其他没有一个是由 GConf2 组件所提供的，因为里面的文件都是由各应用程序所自行定义的，并且都以 XML 的格式存在。这些文件的目的，当安装该应用程序时，可以将其自行的键值写入到注册的目录中，往后启动时提供 GNOME 参考使用。

◆ gdm

gdm 目录的全名为 GNOME Display Manger，也就是协助 X Window 启动的管理软件（如 X Window 登录的方式），如果各用户以为所有进入 X Window 都会用到这一个目录，那就错了！因为当用户直接使用【startx】命令进入 X Window 时，是不须要经过 GDM 的过程（所以就看不到登录的画面，会直接以执行 startx 的用户进入 X Window）的，一般是因为启动时会以【init 5】(因为/etc/inittab 中的定义）进入 X Window，所以才会参考到此目录中的配置。

在 GDM 中主要配置文件为“custom.conf”文件，在 X Window 下可以利用【gdmsetup】命令对这个文件进行配置，如自动登录、登录时的画面、远程登录限制等，都是在这个配置文件中所定义的。在 X Window 参考完主要的配置文件后，就会开始启动登录的程序供用户进入 X Window。

在登录之后，这个目录也定义在登录后一直到离开 X Window 时各阶段所做的事情(如图 6-92 所示，总共分为三阶段)，其执行的顺序依次为：

Ⓐ PostLogin：在 X Window 的登录画面出现后，用户开始进行登录时执行，也就是当登录时用户可以利用这一文件，进行一些默认的配置，但和接下来的 Presession 阶段一样，这时候尚未进入 X Window 的主画面，所以有些部分是无法执行的。

Ⓑ PreSession：gnome-session 是提供用户 X Window 接口的程序（也就是通过 gnome-session 才可看到 GUI 的操作接口），而 PreSession 是介于 PostLogin 与 gnome-session 所执行的，可以想象得到，因为登录看到画面的时间原本就很短，而 PostLogin 与 PreSession 的时间都在登录与看到 X Window 画面之前，所以中间的时间是很短暂的。

Ⓒ PostSession：PostSession 所执行的时间在 gnome-session 执行完成后，和用户注销之前。所谓的执行完成，也就是当用户要注销或是 gnome-session 被 kill 掉时的操作，所以这一目录下的配置可以让我们定义一些注销时所需要的操作，如图 6-92 所示。

图 6-92：/etc/gdm 目录下的文件

在此笔者要强调的是，这一种做法是针对 GDM 的，也就是说，Redhat 是以 GDM 为 Display Manager，但不是每一套 Linux 都以 GDM 为 Display Manager，如 SuSE 便以 KDE 为主，所以并不适用于每一个操作系统，但若要使用，只要在 X Window 下将 Display Manager 调整为 GDM 就可以了。

◆ gnome-vfs-2.0

很多用户应该都做过一件事，就是在 Linux 的“文件管理器”（就是“我的电脑”啦）中直接将浏览文件的路径设为对方 SAMBA 服务所需要的格式，如“smb://192.1.1.1/share”，对于不熟悉 Linux 的用户来说，可以想成和 Windows 下的文件管理器，把路径设为网上邻居（//192.1.1.1）的方式是一样的意思，只是使用的方式和操作系统不同。

但这里的重点，是所呈现的文件列表结果，和一般正常文件系统（如 FAT、NTFS、EXT2 或 EXT3）中的文件及目录是一样的，不需要额外使用其他软件或多做配置，原因是系统会有一套虚拟的文件系统（Virtual File System，但这里的 VFS 和系统底层的 VFS，如 sysfs 是不一样的东西）帮用户自动转换为可直接读取的方式，所以在 GNOME 下面就有一套 GNOME VFS 的机制，也就是这一目录主要提供的服务。

在这一目录中的“modules”子目录，有各种文件系统的配置（如图 6-93 所示），让 GNOME（也就是进入 X Window 后）的系统可以知道每一种文件格式要如何开启或浏览，而所有的配置都需要有相对应的函数库，像刚刚的例子“SAMBA”，就需要有 libsmb.so 的函数库存在。往后如果希望 GNOME 可以支持多一点的格式，只要在这个目录中做修改就可以了，但当然要有一个可以对应的文件系统函数库。

图 6-93：GNOME VFS 的主要配置文件与函数库

有一点要补充的，就是目前 GNOME 已经有 GNOME VFS 的替代方案，称为 GVFS，有可能在这一两年就全面换为这一种新的虚拟文件系统格式，然而，也要等所有软件都转换到 GVFS 后才会全面淘汰，所以应该还需要一段时间，目前使用 GNOME VFS 仍没问题。

◆ gtk 和 gtk-2.0

gtk 是由【gtk+】组件所提供的目录，也是 Widget 的一种组件，X Window 窗口的画面都会有一些颜色、按钮或图案，这些友善的接口，包含该软件选项的画面、选项的按钮、滚动轴的样式等，都需要有一个图形的函数库，也就是所谓的 Widget 来提供。您可能没看过 Widget 这个名词，但这里的 GTK 就是 widget 中的一种。以目前来说，GNOME 下使用的是 GTK+，而 KDE 下

则是 QT。以 Fedora 为例，GNOME 的用户在安装某些软件或使用 X Window 时，只要在 GNOME 下面执行的软件，都是使用 GTK+的；换句话说，若用户要以 KDE 为默认的 Desktop Manager，就会变成全部使用 QT，因此，这两种 Widget 在 GNOME 与 KDE 间是不能够混用的，因为当初开发的程序就已经决定要用 GTK+还是 QT。

◆ kde

虽然 Fedora 中 X Window 默认使用的 Desktop Manager（桌面环境）是 GNOME，但一样可以换成 KDE 的接口，而接口在更换成 KDE 后，当然就要由原本的 GNOME 配置，替换为 KDE 的配置，而这一个目录就是 KDE Desktop Manager 的主要配置目录（如图 6-94 所示）。

```
[root@LinuxTree kde]# ls
[root@LinuxTree kde]# ls -l kdm/
total 64
-rw-r--r-- 1 root root   360 2007-05-16 04:55 backgroundrc
-rw-r--r-- 1 root root 21487 2007-05-21 17:45 kdmrc
lrwxrwxrwx 1 root root    21 2008-02-16 10:10 Xaccess -> ../../X11/xdm/Xaccess
lrwxrwxrwx 1 root root    24 2008-02-16 10:10 Xresources -> ../../X11/xdm/Xresou
rces
lrwxrwxrwx 1 root root    22 2008-02-16 10:10 Xservers -> ../../X11/xdm/Xservers
-rwxr-xr-x 1 root root   207 2006-03-30 03:14 Xsession
lrwxrwxrwx 1 root root    22 2008-02-16 10:10 Xsetup -> ../../X11/xdm/Xsetup_0
lrwxrwxrwx 1 root root    22 2008-02-16 10:10 Xwilling -> ../../X11/xdm/Xwilling
[root@LinuxTree kde]#
```

KDE Display Manager
主要的配置文件

图 6-94：kde 目录下的文件

像 KDE 主要的 Display Manager 配置部分，在/etc/kde/kdm 子目录中，其配置文件内容包含登录的操作、登录方式及其他 kdm 的运行细节配置，都可以在这个文件下配置，以 kdmrc 文件为主，但大多是采用链接的方式，直接套用 xdm（Display Manager 中的一种，GNOME 使用的是 gdm）的配置目录。

另外，除了 Display Manager 的配置之外，在/etc/kde 目录中，还有一些环境、关机及 X Window 共享接口的配置（这是由 XDG 组织所规定的），但其实 xdg 的目录也是链接到 XDG 的主要配置目录“/etc/xdg”。

◆ NetworkManager

在 Windows 上使用无线网络，不论在任何地方，只要是曾经登录过的无线 AP，系统就可以记录下来，往后只要再经过当地，需要无线网络时，就可以凭这些记录直接登录进去，而不需要任何的操作。Linux 现在也可以如此，GNOME 所推出的 NetworkManager，其目的就是要让用户不需要做任何操作或配置，就可以享受网络的方便，而这一个目录的主要目的，就是可以存放这

些 AP 或细节的信息。这个软件对一些将 Linux 安装在 Notebook 上的用户，应该会有极大的帮助，如果 Linux 上面要使用无线的操作，那真的是有点太多了，使用 NetworkManager 可以大幅降低这些瓶颈。

◆ pango

Pango 是一套协助 GTK+（GNOME 的 Widget Toolkit）将字体描绘出来的函数库，不论任何的字体或语言，都可以通过 Pango 描绘出来，在 Pango 所附的文件中也有一些示例（如图 6-95 所示）。

```
[root@LinuxTree ~]# tail -5 /usr/share/doc/pango-1.16.4/HELLO.txt
Difference among chinese characters in GB, JIS, KSC, BIG5:
 GB      --      元气    开发
 JIS     --      元気    開発
 KSC     --      元氣    開發
 BIG5    --      元氣    開發
[root@LinuxTree ~]#
```

图 6-95：中文字体在不同语言中表现

◆ rhgb

系统在进入 X Window 之前，有一个前置配合的图形接口（如图 6-96 所示），这就是所谓的 rhgb（redhat graphical boot），换句话说，这个功能的主要目的是让系统变得漂亮，而并不是把原本的启动方式取代掉（即使是这一个图形接口，也只是先占用第八个 Console，或者称为 Virtual Terminal 8，这是为了避免占用原本 X Window 默认的第七个 Console）。

至于此目录存在的目的，是因为 rhgb 在启动及关闭的这段时间内，会需要一个临时的空间做存取的操作，如系统和 rhgb 的交互，或者是目前的状态等，因此就将【/etc/rhgb/temp】目录作为此用。当 rhgb 结束时，这里面的文件也会被清空，所以默认【/etc/rhgb/temp】目录下仍是空的。

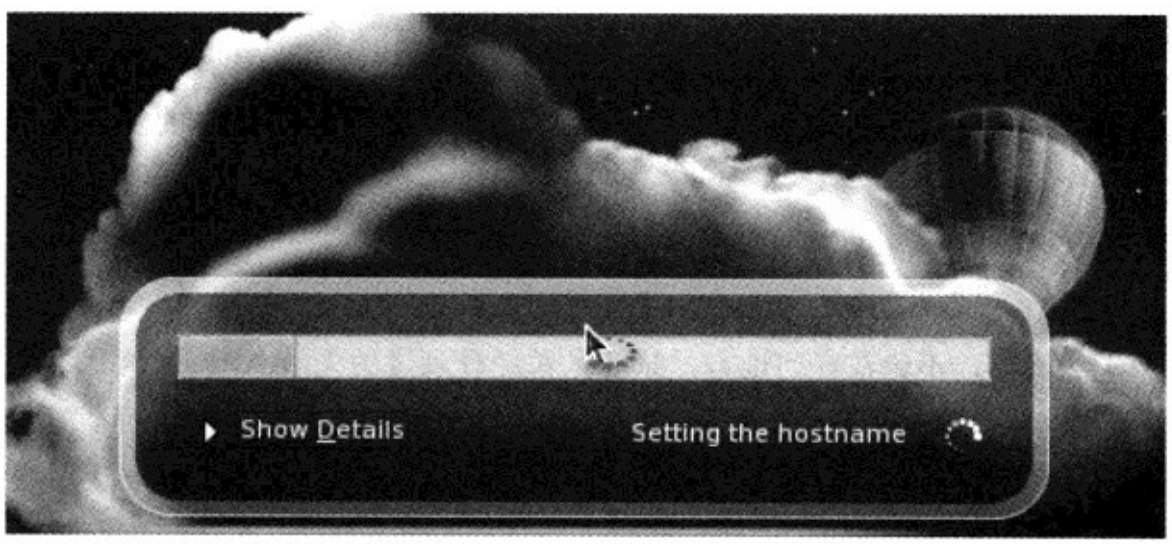


图 6-96：rhgb 在启动时的画面

◆ scim

SCIM（Smart Common Input Method）是 Linux 下目前一种很好用的输入法，比之前的 XCIN 方便多了，同时也支持多国语言，目前 Redhat 的系统默认就采用 SCIM 为标准的输入法接口，不过虽然比 XCIN 好用，但还是很容易产生问题（比如说无法切换输入语言、某些输入法会当掉之类的问题），看来还需要一段时间的改进。

◆ sound

GNOME 下有许多的应用软件，很多都会有其特殊的声音，因此，有一个声音专属的配置目录在【/etc/sound】中，里面有所有声音的命令路径（如图 6-97 所示），但笔者到现在从未听过由 Linux 放出的声音，因为没有一个厂商会愿意在 Server 上多加一张声卡吧。

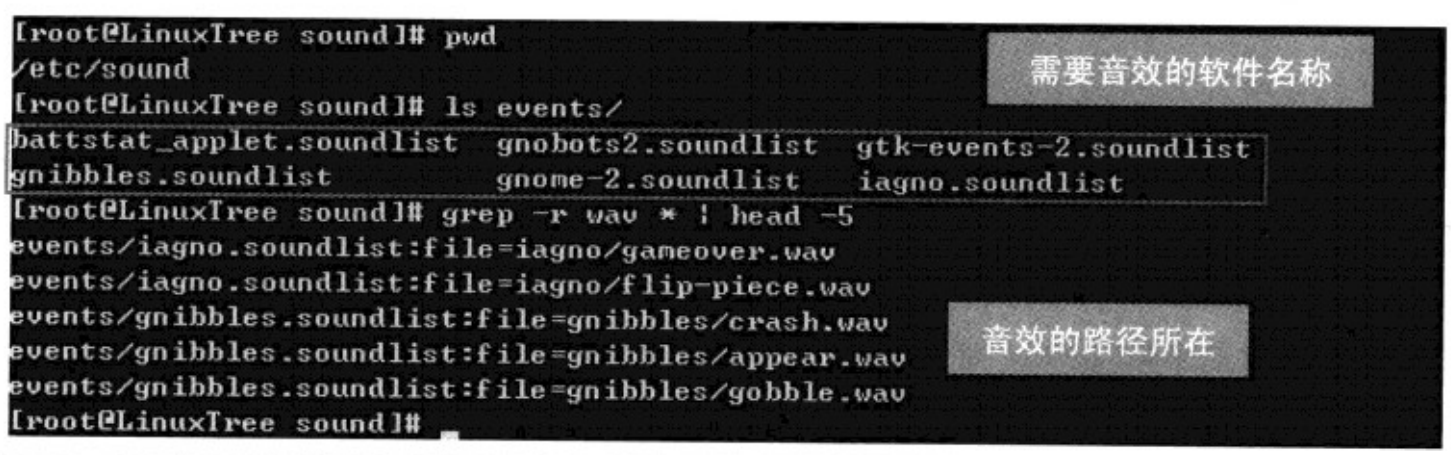


图 6-97：/etc/sound 中部分的声音文件

◆ X11

喜欢使用 X Window 的读者不能不知道这个目录，因为它是 X Window 的核心配置目录，主要的配置文件【xorg.conf】也在其中，除此之外，还有几个比较重要的文件或目录供读者参考。

- prefdm 文件：这是一个判断 X Window 使用哪一个 Display Manager 的文件，如果觉得在哪看过这个文件，那应该就是在【/etc/inttab】文件的最后一行（如图 6-98 所示），当系统进入 runlevel 5 的时候，除了与 runlevel 3 的执行服务不同之外，最后的重点就是要通过这一个文件，决定启动 X Window 的接口，这也是为什么在 shell 下执行【startx】与执行【init 5】有差异的主因。

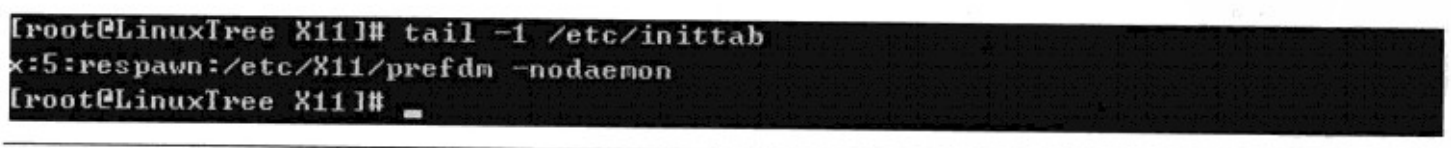


图 6-98：/etc/inittab 中记载有关 prefdm 的部分

- xorg.conf 文件：这就是 X Window 的主要配置文件，在这个文件中可以定义 X Window 所须使用的键盘、鼠标、屏幕、显示卡等相关的硬件设备，但有一点和之前的不太一样，就是字体已经从主配置文件中移除，这样的好处就是系统在进入 X Window 的时候，不会因为字体的问题而卡住（常看到的 Unix/:7100 的信息），减少问题的发生。

  然而，这个文件还有一个常会遇到的问题，就是显示卡驱动程序的配置，细节就不多说，只要提醒读者，当 X Window 无法正常进入，有很大的原因是其中的驱动程序参数不对，这部分可以查看该模块的 man page，大部分都会有所有参数的列表，应该就可以解决显示的问题（在图形接口下似乎没有配置参数的灵活性）。

- xinit 目录：这个子目录中都是一些 X Window 资源相关的配置，如 client、session、键盘、输入法等都在这个目录中有定义。

◆ xdg

X Window 上的菜单画面，就是从这里出来的，所有在 X Window 中使用的菜单文字及分类，都可以在这个目录下做配置，如其中一个子目录【menus】，用户可以通过里面的文件，自定义所有应用程序、系统管理、外观等菜单中的内容，包括有哪些电脑游戏可以在这里配置，对想要自行定义 X Window 菜单画面的用户非常方便。其配置文件结构的设计也非常简单，它是以阶层性的方式排列的，看到后就可以知道如何做配置，如图 6-99 所示。

```
[root@LinuxTree xdg]# ls
                  user-dirs.conf  user-dirs.defaults
[root@LinuxTree xdg]# ls menus/
applications.menu          kde-information.menu      server-settings.menu
                           kde-screensavers.menu     settings.menu
documentation.menu         kde-settings.menu         start-here.menu
gnomecc.menu               preferences.menu          system-settings.menu
gnome-screensavers.menu
kde-applications.menu
[root@LinuxTree xdg]#
```

图 6-99：/etc/xdg 下的文件

6.1.6 其他目录

因为在【/etc】下的子目录类别实在太多，有很多是一些常用软件或是无法归类在前几项的分类中的，读者应该都可以在这里找到。

◆ a2ps

这一个目录中所存放的配置文件，用于将一份文件格式转换为 PostScript 的格式，在某些打

印机或要将文件输出成一份标准格式的文件时，它就会被用到。但现在使用的机会好像很少了，因为 Office 快要变成一种事实上的标准，只有【.doc】文件才是大家都会用的。

不过比较奇怪的是，虽然有一个 a2ps 的配置文件目录，但主要的 a2ps 配置文件，都放在/etc 的下面（如图 6-100.所示），反而 a2ps 目录是一个空目录，不知道是不是以后才把这些文件移进 a2ps 的目录中。

```
[root@LinuxTree etc]# find | grep a2ps
./a2ps
./a2ps-site.cfg
./a2ps.cfg
[root@LinuxTree etc]#
```

图 6-100：放在/etc 下面的 a2ps 配置文件

◆ alsa

alsa 的全名为 Advanced Linux Sound Architecture，是一个专门从事声音处理的团体，主要的任务在于提供 Linux 声音及声音的功能，并试着让其性能达到最佳化。然而，这个团体主要是由 Redhat 及 Novell（目前已为 SuSE 的母公司）所组合而成的，因为看其官方网页的成员列表中，主力是来自这两间公司的核心人员，因此大公司是有其优点的，如图 6-101 所示。

Alsa Team

Core Team

Jaroslav Kysela <perex _at_ perex _dot_ cz>
general hacker, founder of the ALSA project
full-time work allowed through [Red Hat, Inc.] - part of team for 'Kernel drivers'

Takashi Iwai <tiwai _at_ suse _dot_ de>
general hacker
full-time work allowed through [Novell, Inc.] - part of SUSE Labs

Clemens Ladisch <clemens _at_ ladisch _dot_ de>
general hacker

James Courtier-Dutton <James _at_ superbug _dot_ co _dot_ uk>
Sound Blaster drivers, occassional hacker

图 6-101：ALSA 团队的成员列表[3]

◆ awstats

awstats 的全名为 Advanced Web Statistics，是高级的网页流量统计软件。这是一个非常好用的 log 分析软件的配置文件目录，而且使用上非常方便，除了默认的网页记录分析功能外，也可

[3] 图片来源：*http://www.alsa-project.org/main/index.php/Alsa_Team*。

以针对 streaming、FTP、Mail 等服务的记录做分析，并制作图形化的报表。

启动的方式十分简单（当然前提是已经安装 awstats 软件，因为不是每一套操作系统都会安装它），只要在 awstats 的主目录中（在 Fedora7 为/usr/share/awstats/tools）执行【awstats_configure.pl】命令，就可以进行一些基本的配置（要高级或是针对 FTP 等额外服务的配置，就请交给 google 处理），如某配置名称、配置文件路径等，配置完成后，就可以启动了（每次更新记录可在同一目录下使用【awstats_updateall.pl now】命令，如果要定期更新，可以利用 cron table 的机制，定期执行该更新命令）。

从图 6-102 中可以看出，网页中的内容多不胜数，统计及分析得非常详细（统计部分则要归功于 apache 的机制了），因为笔者的系统不是用做网页服务器用，所以自然记录不多（只是为这目录而开罢了），但主要的调整方式还是在此介绍的【/etc/awstats】中的文件做配置。

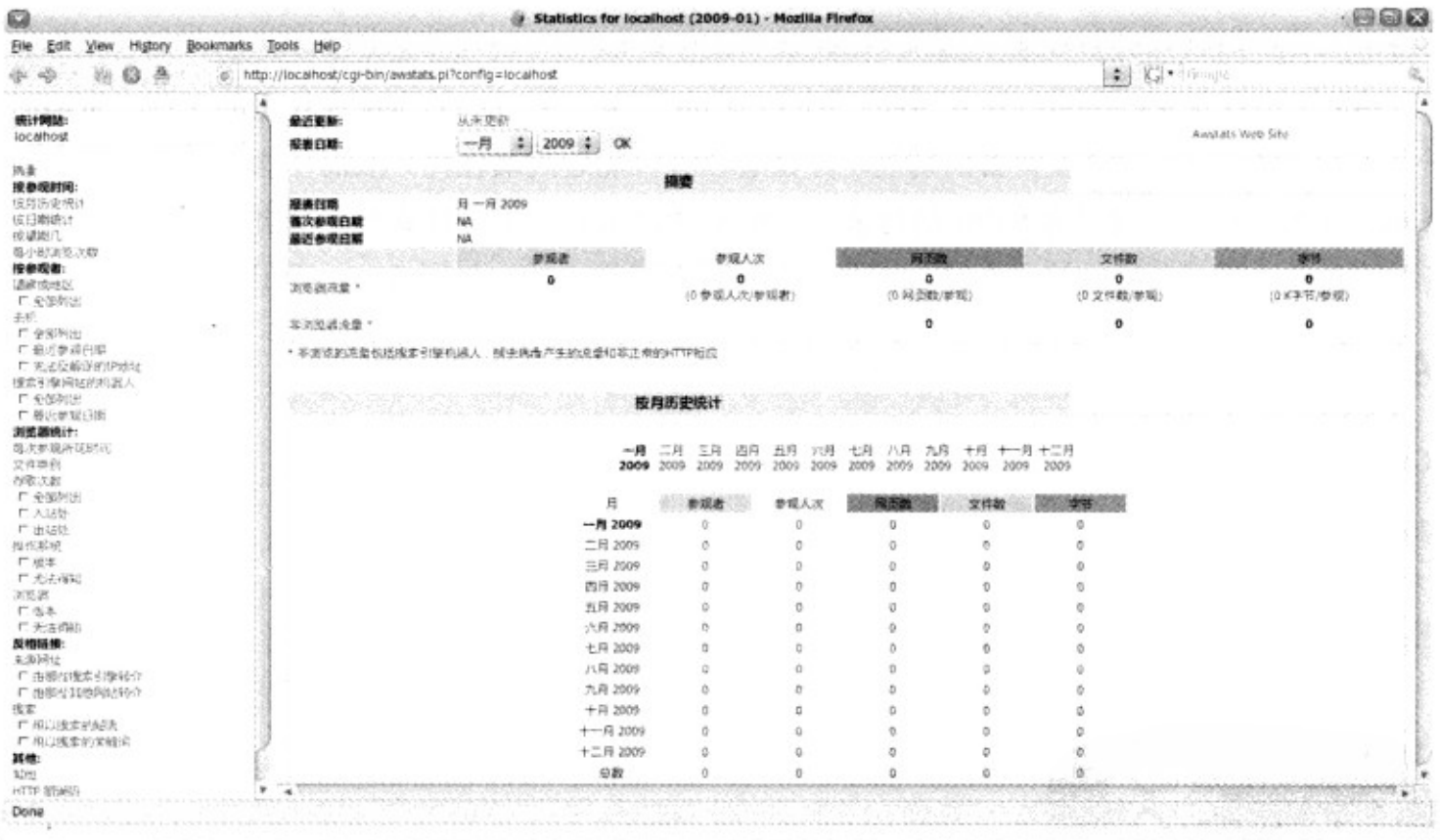


图 6-102：awstats 的网页画面

默认应该是英文，笔者将“/etc/awstats/”中的主配置文件的参数“Lang”修改成“tw”的值，就可以用中文了。

◆ festival

这个目录是 festival 软件的配置文件目录，实在太好玩了！其实也不能说是这个目录好玩，而是 festival 软件的目的比较特别。因为 festival 软件的目的就是要将“文字翻译为声音”，虽然

好像不是非常实用的一件事，但实在是很好玩的一个功能。festival 的组件中有一个独立的程序为【text2wav】，可以帮用户将文字通过【festival】命令与机制，转换为实体的声音文件（wav 文件）。

因为在文字上无法呈现所产生的声音，但读者可以通过图 6-103，看出 text2wav 利用 festival 将文字转换为声音的过程，实际上听起来还真的很清楚（听到的 test.wav 文件声音就是原本的文字 “Hello, I am here. Can you here me?”），应该可以用来做一下语音服务之类的应用。

```
[root@LinuxTree tmp]# cat test.txt
Hello I am here, can you here me?
[root@LinuxTree tmp]# text2wave -o test.wav test.txt
Duration tree extreme for r 3.64693
[root@LinuxTree tmp]# file test.wav
test.wav: RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 160
00 Hz
[root@LinuxTree tmp]#
```

图 6-103：用 festival 将文字转语音实现示例

◆ ghostscript

在 Linux 下要读取 Adobe 格式（如 PostScript 或 PDF 格式）的文件，最方便的方式就是使用 gs（ghostscript）命令，但显示时会使用何种字体作为默认字体，就在这个目录中做配置。在这个目录中的文件（如图 6-104 所示），大都是字体的配置文件，默认也都以现有已安装好的字体为主，若须要变更字体，只要将文件内的字体路径变换掉。

```
[root@LinuxTree ghostscript]# pwd
/etc/ghostscript
[root@LinuxTree ghostscript]# ls
            cidfmap.zh_CN  CIDFnmap.zh_CN  FAPIcidfmap.ko      FAPIcidfmap.zh_TW
cidfmap.ko  cidfmap.zh_TW  CIDFnmap.zh_TW  FAPIcidfmap.zh_CN
[root@LinuxTree ghostscript]#
```

图 6-104：/etc/ghostscript 目录下的文件列表

◆ gimp

GIMP 在 Linux 下是赫赫有名的绘图软件，网络上常常可以看到有人将这一套绘图工具和 Photoshop 做比较，这个目录就是这一套软件的主要配置文件所在，但每一个用户会在其主目录下，有另一份针对个人的 GIMP 的相关配置与目录，在第一次启动时会建立。

◆ gre.d

GRE 是 Mozilla 注册的一种机制（Mozilla 在 Windows 中注册的方式就是使用机码注册），主要在告诉 Mozilla 的软件，当执行时可以马上知道所要执行的是哪一个 Mozilla 的软件。因此，在 gre.d 的目录中（如图 6-105 所示），会直接在 gre.conf 文件中注明所使用软件的版本和路径。

```
[root@LinuxTree gre.d]# pwd
/etc/gre.d
[root@LinuxTree gre.d]# cat gre.conf
[2.0.0.3]
GRE_PATH=/usr/lib/firefox-2.0.0.3
[root@LinuxTree gre.d]#
```

图 6-105：gre.d 目录中的配置

◆ htdig

htdig 是一个网页搜寻引擎的软件，由“ht://Dig”公司所提供，主要目的是让用户可以通过网页的窗体，以 HTTP 的协议对单台或多台网站进行文字的搜寻，而它可以搜寻的有 HTML 与 TXT 两种格式的文件，主要的配置文件为 htdig.conf。不过，因为主要是以网页搜寻为其目标，所以要先有网页的表格产生后才能进行搜寻，而不是在一般 console 下所使用的搜寻工具，这和下面提到的 namazu 工具程序是不一样的。

有兴趣或有须要开发搜寻接口的读者可以参考以下的 htdig 官方网页，里面有对 htdig 很完整的信息。

http://www.htdig.org/

◆ iproute2

iproute2 是一套非常强大的网络管理软件，在以前，很多的网络命令都分开使用，如 ifconfig、route、arp 等常用到的网络命令，其实都可以由 iproute2 所提供的【ip】命令所取代，这也是当初的目的（但其实凭良心说，笔者也还是在用旧的命令，懒得换掉，真须要用 iproute2 时才会使用，这应该是一种美丽错误吧）。iproute2 提供的功能有很多种，而【/etc/iproute2】目录则是存放一些网络的基本配置值（如图 6-106 所示）。

比如，目录中的“rt_dsfield”文件，是在使用带宽管理功能时（iproute2 所提供的 Traffic Control 流量控管，也就是“tc”命令），其中一种 Differentiated Service（差异性服务）作法上，其数据包表头中优先权字段的所有数值。

◆ java

这个目录是由 jpackage-utils 组件所提供的，也是这个组件的主配置目录，除 java 目录外，在/etc 下另外还有【maven】、【jvm】及【jvm-common】的两个目录也都是由 jpackage-utils 所产生的，这里仅列出主要的【/etc/java】目录。JPackage 是一个专门为了提供 JAVA 程序与函数库所存在的软件，JPackage 的好处有以下几点。

```
[root@LinuxTree iproute2]# ls
ematch_map  rt_dsfield  rt_protos  rt_realms  rt_scopes  rt_tables
[root@LinuxTree iproute2]# cat rt_dsfield
0x10    lowdelay
0x08    throughput
0x04    reliability
# This value overlap with ECT, do not use it!
0x02    mincost
# These values seems do not want to die, Cisco likes them by a strange reason.
0x20    priority
0x40    immediate
0x60    flash
0x80    flash-override
0xa0    critical
0xc0    internet
0xe0    network
[root@LinuxTree iproute2]#
```

图 6-106：/etc/iproute2 目录中的文件及示例

- 使用兼容于 Linux 的标准，让用户更方便，如 FHS。
- 可管理多版本的 JAVA 程序，这对各种应用程序来说是十分有利的，因为每一种应用程序所需的 JAVA 版本可能不一致，通过 JPackage 可以有效地管理所有版本。
- 建立 JAVA 软件在包装及安装上的规则。

◆ libvirt

这是一套管理 Linux 下虚拟机（Virtual Machine）的工具程序，里面存放一些管理方面的配置文件，但并不是存放虚拟机的配置文件，而这一套管理工具可以支持几种虚拟机的种类，在这里列出 libvirt 所支持目前较常看到的三种 Linux 虚拟方式。

- XEN

（http://www.cl.cam.ac.uk/research/srg/netos/xen/index.html）

目前 Redhat 及 SuSE 在安装时（RHEL 需要序号）就已经将 XEN 当作默认的虚拟机软件，但 XEN 的做法需要 kernel 的配合，所以使用 XEN 时，会发现启动时多一个“XXX-xen”的 kernel 选项，必须要用该 kernel 启动才可以使用 XEN。

- QEMU

（http://fabrice.bellard.free.fr/qemu/）

大部分的 Linux 默认也都会安装这一虚拟机软件，因为 QEMU 是直接模拟出整个系统的硬件（包含 CPU），因此在很多的平台上都可以支持，但也因为模拟的程度太广而造成性能上的降低，因此目前比较少默认会以 QEMU 为主要的虚拟方式。

- KVM

（*http://kvm.qumranet.com/kvmwiki*）

KVM 是一种以 kernel 为根本的虚拟解决方案，只针对“全模拟”的虚拟方式为主，因此，若要使用 KVM，CPU 就一定要支持 Intel 的 VT 或 AMD 的 AMD-V 的技术。但由于 KVM 的使用方式是以模块（kvm.ko）存在于系统中，操作上非常简单（甚至比 XEN 还简单，至少不需要使用到第二种 kernel），因此，对需要轻巧型虚拟方式的用户来说是最好不过了，笔者就很喜欢这样的方式，因为这是很单纯的一种模式。

◆ mgetty+sendfax

想要用 Linux 架构一台 Fax Server，就可以好好看一下这个目录，在这个目录中，可以利用 mgetty 来配置许多有关传真接收与发送的操作，甚至可以配置朋友的电话号码，不过秉持好用为主的笔者，会在 Windows 下直接安装一个 WinFax 之类的软件（Linux 爱好者请不要打我）。

◆ mono

Mono 是一套由 Novell 所推出的软件，还记得前一阵子 Novell 与微软合作的新闻吗？没错，Mono 这一套软件就是合作下的产物，图 6-107 是 linux-watch 新闻中的小片段，它说明了 Mono 软件是这两间公司合作下所产生的第一个产物，这也是 Linux 的一个重要里程碑，因为 Mono 代表着可以在 Linux 下开发或执行【.NET】的 Client 和 Server 端软件。

Nov. 09, 2006

The first fruits of Microsoft's and Novell's new-found collaboration were on display Nov. 9 in Barcelona Spain, where at a Microsoft TechEd Developers Conference & Expo. Novell introduced Mono 1.2.

图 6-107：有关 Mono 软件的新闻片段[4]

除了 Mono，还有哪些东西是值得我们期待的？以下几点是目前知道已列入他们的合作范围的：

- 双方专利的共享。
- 虚拟化技术与其管理技术的共通性。
- 发展 Open Office 和 MS Office 共通的文件标准。
- Mono、OpenOffice 和 Samba 开发组将可在合约期间参考微软的技术。

◆ namazu

这是针对一套 namazu 程序所存放配置文件的目录，namazu 是一套搜寻程序，和 find 命令的

[4] 参考来源：*http://www.linux-watch.com/news/NS9711352426.html*。

不同之处在于，namazu 的主要目的在帮系统先行建立某目录下的索引文件（index），并且只将【.html】及【.txt】的文件记录在其中，因为它原本的目的就是为了网页搜寻所使用的。另外，和之前提到的 htdit 不同之处则是，这一套是以本机搜寻网页文件为主要功能，而 htdig 则是以搜寻网络上不同网站中的内容为主要目的。

基本上，namazu 属于一种单机网页搜寻软件，并且它可以让用户在 Linux console 下直接操作，当作搜寻软件使用。举例来说，若用户希望在系统默认的网页目录（/var/www）下搜寻文件，可以利用 namazu 组件的【mknmz】命令，先帮/var/www 目录建立其专属的索引文件（如图 6-108 所示），mknmz 在执行命令时，会将所有的索引文件数据全记录在执行的目录中（读者若有兴趣，可以看一下里面的 NMZ.r 文件），所有文件名都会以 NMZ 为开头。

```
[root@LinuxTree search]# pwd
/tmp/search
[root@LinuxTree search]# mknmz /var/www/
[root@LinuxTree search]# ls
NMZ.body                NMZ.field.uri          NMZ.result.normal.es
NMZ.body.es             NMZ.field.uri.i        NMZ.result.normal.fr
NMZ.body.fr             NMZ.foot               NMZ.result.normal.ja
NMZ.body.ja             NMZ.foot.de            NMZ.result.normal.pl
NMZ.body.pl             NMZ.foot.es            NMZ.result.short
NMZ.err                 NMZ.foot.fr            NMZ.result.short.de
```

图 6-108：建立欲搜寻目录的索引文件

有了这些索引文件，之后要搜寻任何字符串，就可以利用这个目录来进行搜寻（如图 6-109 所示），最方便的是，搜寻的结果中，还会有针对每一条数据的排名和评分，有点像 google 的特性（只是不知道标准是什么），感觉比 find 或 grep 所找出来的结果更准确（如果会有很多条数据）。

另一个重点，因为这是专门针对 Web 网页所搜寻的程序，所以在图 6-109 中也可以看出，其结果是以网页的标头（title）显示其排名（当然要 HTML 的文件才有效），这对命令下要找某些网页真的太方便了。

```
[root@LinuxTree search]# namazu -sq administrator . | head -10
1. core - Apache HTTP Server (score: 8)
/var/www/manual/mod/core.html (189,157 bytes)



2. Test Page for the Apache HTTP Server on Fedora (score: 8)
/var/www/error/noindex.html (3,926 bytes)


[root@LinuxTree search]#
```

图 6-109：利用 namazu 所找出的结果

◆ nas

nas 的全名为 Network Audio System（不是网络储存设备的 Network-attached Storage 唷），该软件的功能在于通过网络，播放或录制声音文件、声音文件或音乐等格式，因此，nas 软件自然搭配许多声音相关的命令（如 audial、auedit 或 auphone 等），让用户可以在一般的 shell 接口下操作。所以在这个主要的配置目录中（只有一个 nasd.conf 文件），可以配置许多声音相关的细节操作，以及所须使用到的硬件接口。

◆ openvpn

OpenVPN 是一个 Linux 下的 SSL VPN 解决方案，当企业需要有一个安全的网络（不论是有线或无线）时，就可以使用这一套软件以达到目的，但在没有使用到 OpenVPN 的情况下，默认这一目录只是空的，并不会有任何文件，但用户可以在“/usr/share/doc/openvpn-2.1/sample-config-files/”目录下找到适合的配置文件进行使用，细节的配置可以参考官方网站：

http://openvpn.net/

◆ php.d 与 phpldapadmin

这两个都是与 PHP 相关的目录，但都不是 PHP 的主要配置文件（PHP 主要的配置文件为 /etc/php.ini），唯一的差别，【php.d】存放的是各软件（如 mysql、pgsql 或 ldap 等）与 PHP 相关的配置文件；而【phpldapadmin】则是专门存放“phpldapadmin”管理软件的配置文件。

◆ purple

这个目录因为是 pidgin 软件底层的配置目录，可能会让读者有些疑惑。

Q：什么是 pidgin？

A：就是之前的【gaim】，也就是在 Linux 上要使用 MSN 的一种方案，可以提供多种 IM（Instant Messaging）协议，支持 Google Talk、Yahoo!、MSN、ICQ、AOL 或 QQ 等更多类似的即时通信软件，而且可以同时上线使用，是非常方便的软件。

Q：何谓 pidgin 的底层？

A：这个【purple】的目录是由一套 libpurple 的组件所提供的，这一个组件其实就是 pidgin 所需要的函数库，也就是说，大家所使用的 pidgin 只是一个外表，里面的核心程序其实是 libpurple，不过，若读者看过 pidgin 程序，或许就会知道为什么核心部分要取名为 libpurple，因为这个软件的图案是一只全身“紫色”的鸽子（这是笔者的猜测啦），如图 6-110 所示。

图 6-110：pidgin 的 Logo 图示

◆ reader.conf.d

这是一个储存 smart card（芯片卡）配置文件的目录，由 pcsc-lite 组件（PC/SC Card）所提供，但这个目录并非主配置文件的位置，主要的配置文件是【/etc/reader.conf】文件，该文件中所记录的是卡片阅读机所须使用到的模块，而【reader.conf.d】目录的目的，则在于存放一些需要被【/etc/reader.conf】文件一并加载的信息，所以是附加的目录（并不是直接通过 include 附加进去，而是利用“/usr/sbin/update-reader.conf”文件载入）。

◆ tomcat5

因为现在 JAVA 网页技术当道，Tomcat 已经是一个非常有名的软件，最早由 SUN 公司所推出，再交由 Apache 组织。一般 Tomcat 会搭配 Apache 一起提供网页服务（虽然 Tomcat 本身就有小型的 Web Server 的条件），以及提供 Servlet 与 JSP 执行的环境，对网页程序有兴趣的，不妨参考 Apache 官方网站上对 Tomcat 的详细介绍。

http://tomcat.apache.org/

◆ vdr

这个目录是由 vdr 软件所提供的，光看 vdr 软件的全名（Video Disk Recorder）就知道是一个让大家耳目一新的软件，在 Linux 下竟然可以将电视、卫星频道录下来观赏，这和一般 Linux 给人的感觉完全不同。在这个目录下有许多的配置文件，包含频道的配置、录制的节目、远程控制等的配置，看起来琳琅满目，笔者觉得真是值得一试的软件。

◆ w3m

这是 w3m 软件的主要配置文件，w3m 是一个 Linux 下面 HTML 字符模式的浏览器，和【lynx】

很像的，不过这一只小程序除了基本的 HTML 解析能力外，还有着和【less】一样的分页及搜寻字符串的能力，比之前【lynx】好用一点（并且又支持中文），基本上可以把这只程序想成是【lynx】+【less】的 HTML 浏览器，不过因为是字符模式，好不好看就是另外一回事了。

6.2 /srv

这是一个在根目录下较新的子目录，在比较新的操作系统版本中才看得到，目前连 FHS 都还没有定出如何在这个目录下为子目录命名的原则，唯一可以确定的是，【/srv】中的文件要储存本机所提供的一些配置文件，但至于是哪一类软件或配置，就没有限定，但可由系统管理员自定义。目前在笔者的系统中默认只有一个【www】目录中保存有文件，其他如【vdr】或【audio】都只是空目录，应该是软件的预先规划。

【www】中的目录名称很容易让人想到 httpd 的服务，但其实存放在这个目录中的并非 httpd 服务，而是另一个“lighttpd”服务（如图 6-111 所示）。可能是为了避免混在一起，所以“lighttpd”特地将其本身的首页选择放在【/etc/srv】中，而不是在一般所存放的【/var/www】中，的确是清楚多了。

```
[root@LinuxTree srv]# find www/
www/
www/lighttpd
www/lighttpd/index.html
www/lighttpd/powered_by_fedora.png
www/lighttpd/favicon.ico
www/lighttpd/light_button.png
www/lighttpd/light_logo.png
[root@LinuxTree srv]#
```

图 6-111：/etc/srv 中 www 目录存放的数据

总结

【/etc】可以说是文件数量非常庞大的一个目录，笔者光在这一章就花了非常多的时间，当然也是因为跨越太多领域（一个目录通常代表的就是一个新的领域），所以很多时间都花在了找数据上面。让笔者比较纳闷的是，既然已经有【/etc】的存在，为什么会有【/srv】的产生，因为【/etc】目录原本就可以存放配置文件以外的一些目录文件，实在不必要另外在根目录下再开一个【/srv】的目录，笔者在文件中也没看到其由来，可能要等各厂商都开始配合之后才能感受到差异吧！

因为在这个目录下，每一个操作系统厂商的使用方式都有些许不同，又会因安装软件的数量而异，读者很有可能会看到在本章没有提到的目录或文件，如果有需要，可以来信让笔者有机会再加以补充。

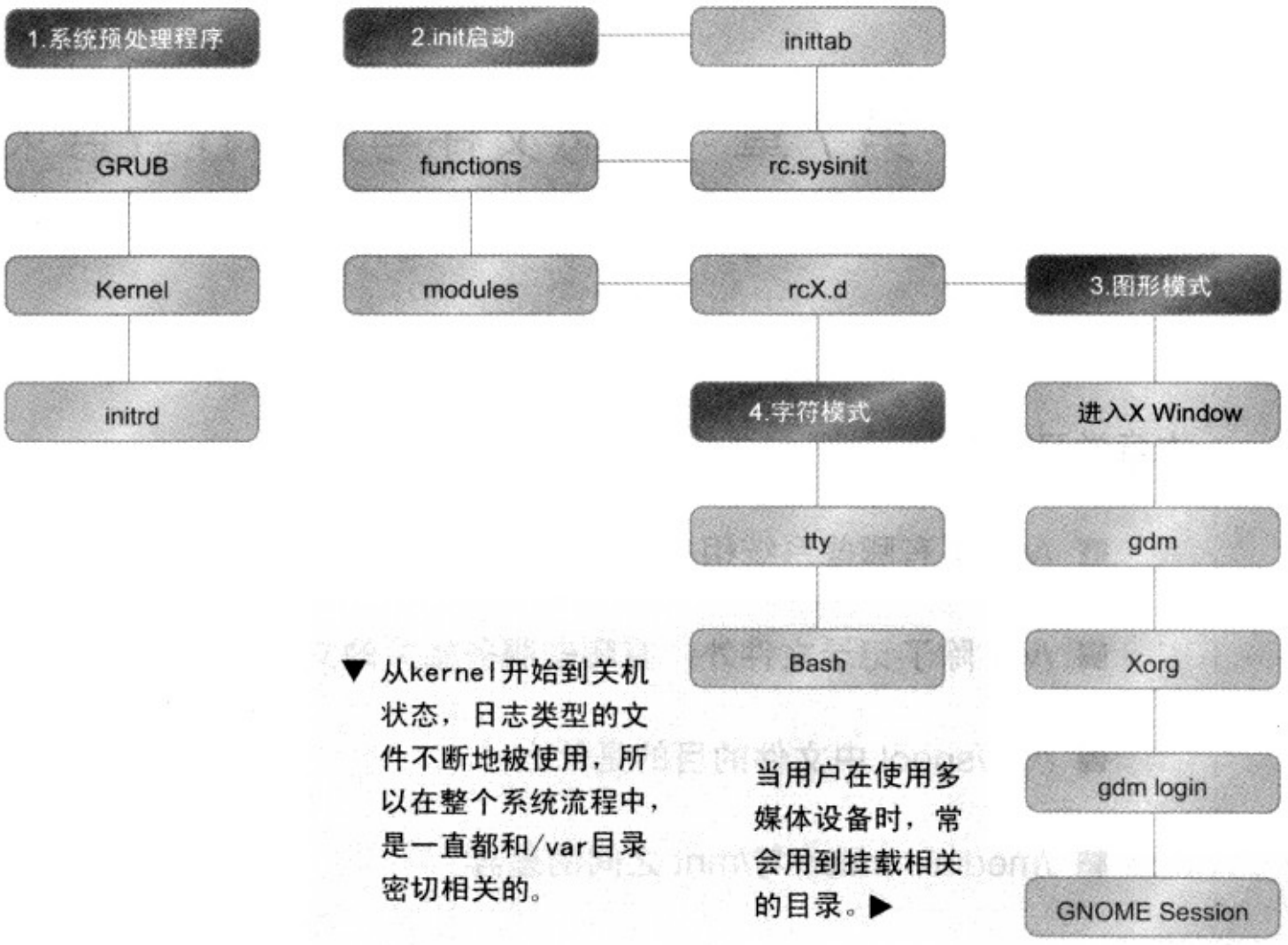
第 7 章　日志文件与媒体挂载目录

本章学习重点

- /var 下有哪些系统相关的记录文件
- /var 除了记录文件外，有哪些服务相关的文件
- /var/spool 中文件的目的是什么
- /media、/misc 与/mnt 之间的差异

系统流程与章节内容对照图

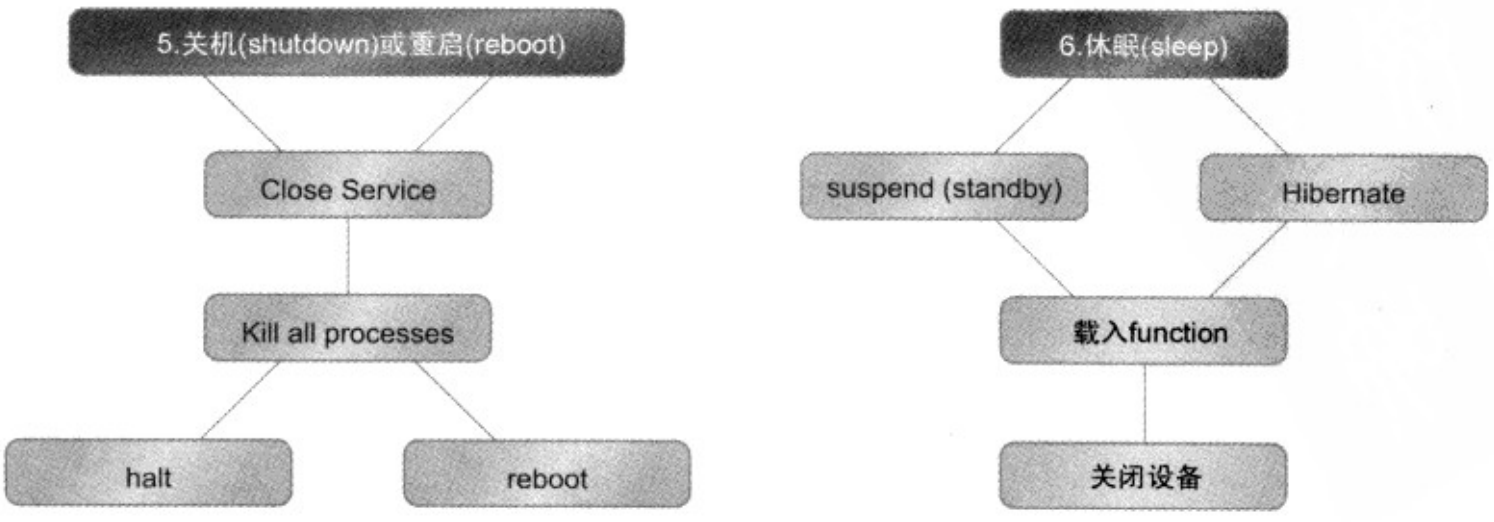
- 启动流程



▼ 从kernel开始到关机状态，日志类型的文件不断地被使用，所以在整个系统流程中，是一直都和/var目录密切相关的。

当用户在使用多媒体设备时，常会用到挂载相关的目录。▶

- 关机流程



日志文件在 Linux 中占有非常重要的角色，不论是系统、软件、服务、网络等相关的记录，都非常详细，只要找对记录文件，就可以分析出所想要的答案，所以，唯一的问题出在哪个记录文件是我们所要的。但是，夹杂在日志文件目录中的，以及一些属于比较特别的文件，由于名称只是笔者自己所区分的一种分类方式，让读者比较好分辨而已，并非只能放置记录文件的文件（只是记录文件是最常用也是最重要的），其他像软件的执行、mail 的源文件、Proxy 的数据等也都在这个目录中，但这些文件的前提，就是都会以不断更新其文件状况的方式存在，这是很重要的。

至于媒体挂载目录，对新手而言特别重要，因为在一开始接触 Linux 的时候，对每一种设备所须使用到的设备文件不熟悉，文件格式也可能分不清楚，在这种情况下，就无法知道刚刚挂载到系统上的 CDROM、DVDROM、USB Key、Floppy 等，要如何与这些硬件交互，也就是对存储这些设备上的文件会有问题。但通过系统所制定的媒体专用目录，当用户挂载上去之后，就固定到该目录下查看，或者在 X Window 下就会直接显示其对应的图标，并可以使用该媒体。

7.1 动态文件记录区【/var】

Linux 系统中最主要的记录文件默认都存放在【/var】目录下，除此之外，此目录还累积了大量的临时文件或供系统确认操作的文件，但也因为这些文件都会随着时间而变动，因此，文件大小都是不一定的。

因为该目录中的文件大小会不断地变动，文件数目也会不断增加，所以一般会单独放在其他的分区中，避免影响到根目录所存在的分区，如果记录文件过大，或者内容数量过多时，有可能会把系统的文件资源（可储存的内容或文件数目）消耗光，这样会造成系统的服务中断或出现使用上的问题，所以分离出去是一个比较好的做法。

7.1.1 /var/account

这是一个在 Linux 下审核机制（组件名称为 psacct）所需的目录，也就是让系统管理员可以监督系统命令被哪一些人使用？使用多久？对 CPU 或内存的影响是什么？因此，是一种非常有用的审核机制，该目录下的文件都是由“init”及“login”的结果所产生出来的，一共可以分为如下四个文件。

◆ wtmp

主要记录所有人员的登录及注销相关信息。该文件实际是由系统储存在【/var/log】下的，

【psacct】组件通过这一文件，可以让系统管理员很清楚地了解，目前有多少人在使用系统（如图 7-1 所示），以及这些人待在系统多久时间，这个审核机制也为管理员提供了相对应的命令，如“last”及“ac”。

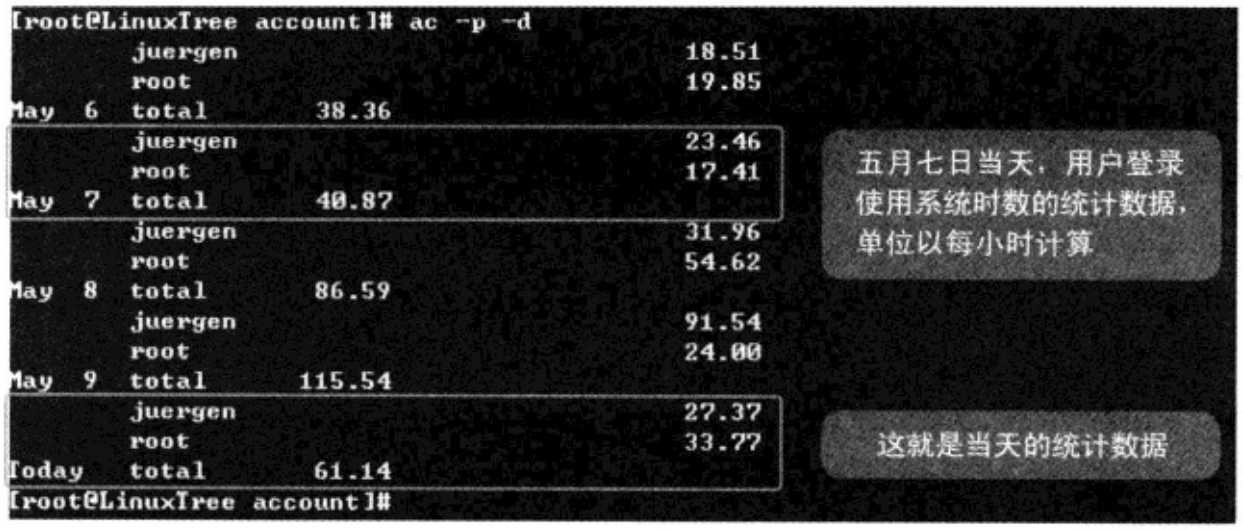


图 7-1：系统用户的使用时间统计表

◆ acct

这一个文件所记录的是每一个命令被使用的记录。不过，在实际的系统下并不一定叫“acct”这个名称，如 Redhat 中则叫做“pacct”，而该文件就是在【/var/account】目录下的唯一实体文件，不过默认是没有启动的，因为用户若检查这个文件，实际上只是一个空壳，没有任何内容在其中。

若用户须要知道命令在系统下的使用记录，可以用【accton /var/account/pacct】命令启动 acct 的功能（如图 7-2 所示），让系统开始记录（不须要重新启动计算机），要检查有没有起始记录，其实只要看文件在使用过任何命令后，文件大小是否开始有变动即可，因为在写入后此文件是无法被直接读取的。接着只须要使用【sa】命令查看结果，就可以得到目前系统与命令的使用关系。

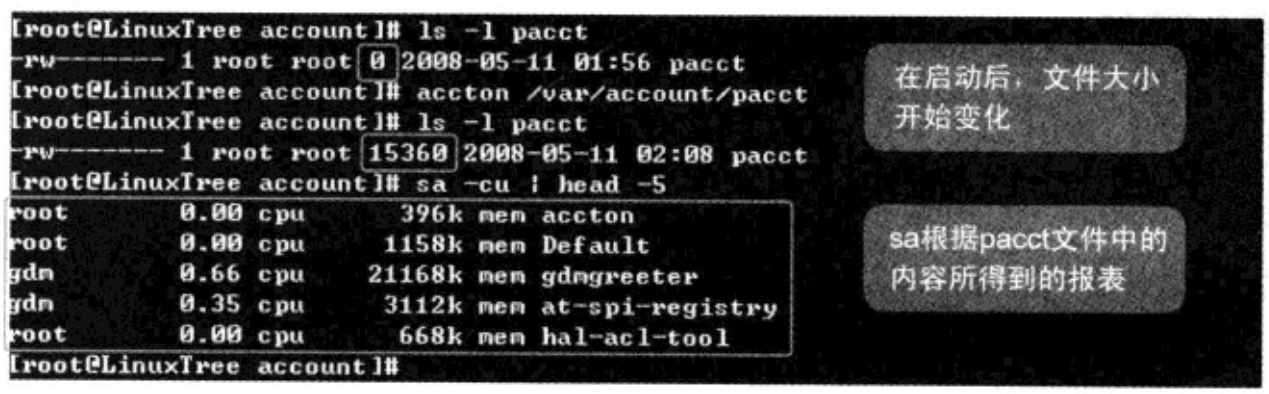


图 7-2：Linux 下命令的使用状况报表

◆ usracct 和 savacct

这两个文件其实是指，如果用户根据以上【sa】命令，按照用户或命令执行所整理得到的结果，将其命名为【usracct】及【savacct】，之后就可以直接依据这两个文件检查用户与命令的使用状况。

当用户在检查【/var/account】目录时，一般都只会看到【pacct】（以 Redhat 为例），并且是一个没有功用的空文件，若须要启动审核的机制，就必须使用【accton】命令，若想要看到【usracct】及【savacct】，可能就需要自己产生了。

7.1.2 /var/cache

该目录下的文件是所有应用程序所产生的 cache data（缓存数据），也就是当应用程序启动时，会将数据留一份在其中。但在该目录下的文件，必须要随时被接受重新产生或还原的文件，并不是永久文件，而且被更新、删除、新增等规则都该由应用程序自行定义，并没有一定的规则。所以，任何在这个目录下的文件都不能当作正式的备份，只供应用程序所使用。

以图 7-3 为例，当用户在查询【ls】命令的用法，使用【man】命令时，就会同时复制一份原始 ls 的 man 文件到此目录下，不过，因为 man 会依不同特性分为好几个段落（如命令使用、系统调用、开发专用之类），并照数字区分开来，所以在 man 子目录下就有相对应的【cat1】到【cat9】的分类目录，即使要找到正确的 cache 文件，也要先确定是在哪一个分类下。

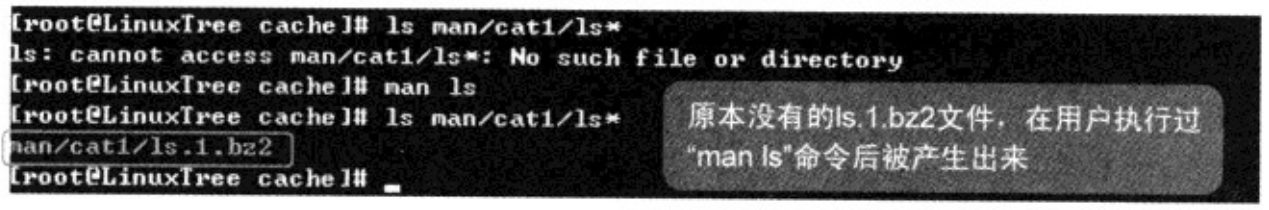


图 7-3：在使用 man 命令后所产生的 cache 文件

7.1.3 /var/empty

目前来看只有 sshd 这一个组件会用到，在 sshd 的 man page 中直接注明（如图 7-4 所示），因为当用户在做 ssh 连接时，提供服务端须要使用该目录下的【/var/empty/sshd】子目录，所以该目录即使是空的也必须存在。不过比较特别的是，这个目录“一定”只能被“root”管理员所拥有，在使用权限发现问题时，是无法启动 sshd 服务的（如图 7-5 所示），这一点是比较容易被忽略的，因为这个目录默认可能是空的，又没有任何配置的问题，所以这是使用 sshd 时要注意的地方。

```
/var/empty/sshd
        chroot(2) directory used by sshd during privilege separation in
        the pre-authentication phase.  The directory should not contain
        any files and must be owned by root and not group or world-
        writable.
```

图 7-4：sshd 的 man page 中有关/var/empty 的相关描述

```
[root@LinuxTree empty]#
drwx--x--x 3 root root 40                    9:48 sshd/
[root@LinuxTree empty]# chown juergen.root sshd/
[root@LinuxTree empty]# /etc/rc.d/init.d/sshd start
Starting sshd: /var/empty/sshd must be owned by root and not group or world-writ
able.
[FAILED]
[root@LinuxTree empty]#
```

原本的目录属性

更改拥有者后产生的问题

图 7-5：当目录权限改变时所出现的错误

7.1.4 /var/ftp

FTP 服务器软件一般默认会将匿名登录的用户引导到/var/ftp 目录下（见图 7-6），所以该目录即为 FTP 的主目录。虽然在 FTP 服务的配置中，可以指定让匿名用户自由读写这个目录下的文件，但要记得一件事，就是目录的权限还是会限制登录的用户，也就是说，这个目录下原本就有的【pub】子目录，看起来是已经可以使用（当登录后就可以看到这个现成的目录），其实该权限是无法让其他用户写入的，并不是配置 FTP 可写入就算数，目录或文件也需要相对应的权限。

至于这个目录下有哪些目录或文件，一切都由 FTP Server 的管理员决定，系统只是先建立了一个默认的目录，接下来的规划，就是管理员自己的问题，所以，在这个目录中并没有任何硬性的规定。

```
[root@LinuxTree ftp]# pwd
/var/ftp
[root@LinuxTree ftp]# ls -l
total 8
drwxr-xr-x 2
[root@LinuxTree ftp]#
```

默认为没有开启一般用户的写入权限

图 7-6：/var/ftp 目录下的清单

7.1.5 /var/gdm

该目录是 gdm（GNOME Display Manager）所使用的目录，里面所存放的是一些系统当前所占用的 console 记录及通过 gdm 所执行的 X Window 记录（如图 7-7 所示），但必须是通过 gdm 窗口的记录才会存放于此。因此，如果不是通过 gdm 所产生的 X Window（如 startx），就

不会有任何信息在这个目录下。

```
[root@LinuxTree gdm]# ls -a
.  ..  :0.Xauth  :0.Xservers  .cookie  .gdmfifo
[root@LinuxTree gdm]#
```

图 7-7：/var/gdm 目录下的文件

7.1.6 /var/lib

所有应用程序正在执行中的状态信息，都可以记录在这一个目录中，该目录下有很多以应用程序名称命名的目录（如图 7-8 所示），每一个子目录下面都是其执行的状态信息，所以【/var/lib】和【/lib】虽然都是用“lib”的字眼，但意义完全不同，不要被这个字眼所混淆。

```
[root@LinuxTree lib]# ls
alternatives  dhcpv6     logrotate.status  ntp          scrollkeeper  vdradmin
awstats       dovecot    misc              pgsql        sepolgen      webalizer
BackupPC      games      mlocate           php          setroubleshoot xen
bluetooth     hal        multipath         pound        spamassassin  xend
cs            hsqldb     mysql             random-seed  stateless     xenstored
dav           htdig      namazu            rpcbind      texmf         xkb
dbus          iscsi      news              rpm          tomcat5
dhclient      libvirt    nfs               samba        vdr
[root@LinuxTree lib]#
```

图 7-8：/var/lib 下的文件清单

7.1.7 /var/lock

这个目录中的文件很特别，如果细看每个文件，会发现其实都是空文件，为什么会这样，是因为每个服务一开始就会先在【/var/lock】目录下（或者子目录）产生一个该服务的空文件，这样做的目的在于，让该服务在启动或停止时，知道目前是否尚有同样的服务在系统中，以避免重复的现象发生，当服务无预兆的状况停止时也有记录可查。当然，如果服务要停止，也须要将该文件一并删除（如图 7-9 所示）。

```
[root@LinuxTree lock]# ls subsys/sendmail
subsys/sendmail
[root@LinuxTree lock]# /etc/rc.d/init.d/sendmail stop
Shutting down sm-client:   OK  ]
Shutting down sendmail:    OK  ]
[root@LinuxTree lock]# ls subsys/sendmail
ls: cannot access subsys/sendmail: No such file or directory
[root@LinuxTree lock]#
```

图 7-9：/var/lock 下文件的变化

有一个服务非常明显地将此目录的作用显示出来，就是在【/etc/rc.d/】下的【rc.local】服务

（在各 run level 目录中，如“/etc/rc.d/rc3.d”的服务名称为“local”），在该文件中默认只有一行（该文件默认为留给用户自行设定使用），就是在【/var/lock/subsys】的目录下产生【local】的文件，也就是先告知系统，local 服务已经启动，且未停止，让之后要在启用 local 服务时可以发现此冲突。

7.1.8 /var/log

对一个系统管理员来说，每天应该都会用到这一个目录，【/var/log】是一个专门存放所有记录文件的目录（如图 7-10 所示），里面记载着许许多多系统、软件、用户等相关的历史记录。因为所有的记录都在这边，所以文件数目也特别多，不过记录文件的存放还是有一点规则可循。

在目录中可以看到和系统直接相关的文件，如 messages 或 X Window 的 Xorg.0.log 文件，会直接放在【/var/log】的目录下；而软件或服务相关的，则在【/var/log/软件名称】的目录中。

但这只是一种习惯，并非是固定的，如 mysql 或 vsftpd 都有记录文件直接放在【/var/log】的目录中，但还是以目录分类的方式比较容易让用户管理。

在一大堆的文件中，有几个记录文件是一定要存在且非常重要的系统文件，也是 Linux 用户一定要知道的，不过，在每一套 Linux 中所使用的文件名称或机制或多或少还是有些差异的，但大致都一样。

Ⓐ lastlog：该文件只记录所有用户最后一次登录的记录，包括登录的用户、port、时间、IP 等，当用户须要确认之前的登录用户，或者登录的时间都可以用【lastlog】命令将该文件的记录打开，但因为其文件是以二进制文件的方式储存的，所以没有办法直接读取这个二进制文件。

Ⓑ messages：里面记载系统所有的信息，但不只是单次进入系统后的信息，是会累积的（除非大小超过单一记录文件的大小才会被切割，若用【dmesg】指令所看到的就是单次启动后的系统记录），每一单次启动后的信息，前面一段是启动 kernel 时所产生的记录，后面就是由系统所执行的 syslogd 信息服务，为系统所记录的信息，当然所有系统运行中的信息也会实时产生。

Ⓒ wtmp：该文件记录所有用户登录及注销的信息，和刚刚的【lastlog】文件一样是以二进制文件的方式储存的，无法直接读取，也必须靠【last】的命令查看其储存的信息。

```
[root@LinuxTree log]# ls
anaconda.log      cron.4      messages.2        secure.2        vsftpd.log.1
anaconda.syslog   cups        messages.3        secure.3        vsftpd.log.2
anaconda.xlog     dmesg       messages.4        secure.4        vsftpd.log.3
audit             exim        mysqld.log        setroubleshoot  vsftpd.log.4
BackupPC          faillog     news              spooler      C  wtmp
bittorrent        gdm         pm-suspend.log    spooler.1       wtmp.1
boa               httpd       ppp               spooler.2       xen
boot.log       A  lastlog     prelink           spooler.3       xferlog
boot.log.1        lighttpd    rpmpkgs           spooler.4       Xorg.0.log
boot.log.2        mail        rpmpkgs.1         squid           Xorg.0.log.old
boot.log.3        maillog     rpmpkgs.2         tallylog        Xorg.1.log
boot.log.4        maillog.1   rpmpkgs.3         tclhttpd        Xorg.1.log.old
btmp              maillog.2   rpmpkgs.4         tomcat5         Xorg.20.log
cron              maillog.3   samba             vbox            Xorg.setup.log
cron.1            maillog.4   scrollkeeper.log  vdr             yum.log
cron.2         B  messages    secure            vdradmin
cron.3            messages.1  secure.1          vsftpd.log
[root@LinuxTree log]# ls mail
statistics
[root@LinuxTree log]# _
```

图 7-10：/var/log 目录下的文件清单

7.1.9 /var/named

如果系统要作为 DNS 服务器，那么这个目录就是这台系统最重要的目录之一。这个目录用来存放 DNS 所负责的网络中所有主机的正查及反查的数据库文件，所以，用户通过这一台 DNS 服务所得到的域名或 IP 地址，都是由这个目录下的文件解析出来的，如果没有这个目录的信息，这一台 DNS 等于是一台没有作用的 DNS。

里面有几个比较常用且重要的文件，都以“named”为文件名的文件（如图 7-11 所示），DNS 域名的管理员大部分都是在修改这些文件，为的就是增删所管理网络中的域名，最重要的有以下三个文件。

- named.ca：记录所有 DNS 主架构中，最上层的 ROOT 主机地址（所谓的 ROOT 就是所有 DNS 服务器在查询时的最后一道防线，也就是最主要的 DNS 主机，大部分在美国）。
- named.local：反查（也就是用户以 IP 查询域名）时，DNS 服务所参考使用的记录文件。
- localdomain.zone：正查（也就是用户以域名查询 IP）时，DNS 服务所参考使用的记录文件。

但因为这些文件的名称都由主配置文件（在/etc 目录下的 named.conf）所配置，所以在不同的配置条件下，文件名称会有所不同，因此，在对照这些文件时，应该以主配置文件中的文件名称为主。

```
[root@LinuxTree named]# ls
chroot  localdomain.zone  named.broadcast  named.ip6.local  named.zero
data    localhost.zone    named.ca         named.local      slaves
[root@LinuxTree named]#
```

图 7-11：/var/named 目录中的文件清单

7.1.10 /var/nis 和/var/yp

/var/nis 和/var/yp 这两个目录会放在一起，因为同时都会由 NIS 机制所使用的目录，只是两者目的完全不一样。

- 【/var/nis】：在此目录中，主要记录所有网络中每一个 Client 的连接记录，每一个用户大约会占 5KB 的大小，从另一方面想，这一目录和系统的目录最好是分开的，否则，如果因为用户过多而导致硬盘容量不足，则会让系统不稳定。
- 【/var/yp】：这是 Linux 的 NIS 服务主要记录文件的放置目录，里面主要记录所有 NIS 须要知道的主机对应数据，和 DNS 的概念类似，所有在 NIS 中找寻另一台主机的数据，都必须通过该目录中的记录与之对应。

7.1.11 /var/run

/var/run 目录中大部分文件都记载着目前系统正在执行程序的 PID（Process ID）值，每一个文件是一个独立的 PID 记录，故不会有相同值的机会，不过，该目录对所有文件也有一定的规则。

- 文件名称：应该以【程序名称.pid】为正式文件名称，从图 7-12 中可以看到，有部分文件的扩展名并没有以 pid 为名。
- 文件内容：正确的内容应该为【PID 值＋换行】，也就是说，如果【xinetd】的 PID 值为 1234，其【xinetd.pid】中的内容就应该为【1234】和【断行】，也就是当用“cat”命令查看其文件时，会显示“1234”，shell 的提示字符会换行。

不过，在该目录中有一个和大家都不太一样的文件，也值得在此一提，即【utmp】，这个文件所记录的是目前谁在使用该系统？该文件是无法直接读取的，必须通过【utmpdump】命令才可看到其中的内容（如图 7-12 所示），因为笔者是由远程的“10.32.15.207”主机以 telnet 方式用“juergen”的用户登录，所以在 utmp 文件中的记录，就会记载该次登录的 PID 及其他相关的信息。

图 7-12：/var/run 目录下的文件及 utmp 的部分内容

7.1.12 /var/spool

一般在 Linux 中看到 spool 的字眼，代表着一个等待被使用文件或数据的区域，这里当然也不例外，你可以看到在这个目录中，都是一些“临时存放的物品”，随时会被用户所提取，如一般所使用的 email。笔者将几个比较常用到的目录列出供读者参考，但主要目的不变。

◆ anacron

anacron 和 crontab 是相辅相成的服务软件（详细请参考第 6.1 节有关 anacrontab 文件的介绍），【/var/spool/anacron】所存放的文件，则是此服务的时间戳（timestamp），也就是开始计时的系统记录（如图 7-13 所示）。

图 7-13：anacron 的 timestamp 文件

◆ at

at 和 crontab 与 anacron 的差别，在于它是以单次计划任务的方式记录用户所需的服务，而【/var/spool/at】目录，主要就是记录每一个计划任务要做哪些事，以及执行后的结果是什么，

并将其记录下来。

如图 7-14 所示，当用户完成新增一条“计划任务”后，在【/var/spool/at】目录就会直接产生一个由 at 帮助用户建立的 script 文件，有兴趣的读者可以看一下里面的内容，其中不只是记录用户所要执行的命令，还包括执行时所需要的环境变量等信息，因为当执行该条计划任务时，可由 at 提供 shell，并不是由当前的 shell 直接执行的，这点要请读者特别注意。

在完成后，会由系统记录执行所产生出的信息（图 7-14 因为笔者的 script 没有产生信息，因此，没有额外的信息），当执行失败时，用户可以通过该文件做检查的操作，等于是 at 的记录文件。

图 7-14：at 在使用时用来记录的文件

◆ cron

crontab 是最常被使用的计划任务软件，详细的配置方式或说明可参考网络上的文章（多到看不完，但其实配置方式很简单），配置方式大致有两种：利用/etc/cron.*目录（只有 root 有权限使用）或直接用【crontab -e】命令编辑计划任务。前者和此目录无关，因为 crontab 会直接执行其目录下的所有配置，但后者（直接用【crontab -e】命令编辑计划任务）则会在【/var/spool/cron】目录下，为该用户产生出计划任务文件（如图 7-15 所示），当时间到了，crontab 也会参考该目录下“所有用户”自行配置好的计划任务，为所有用户带来便利性也是这个目录存在的意义。

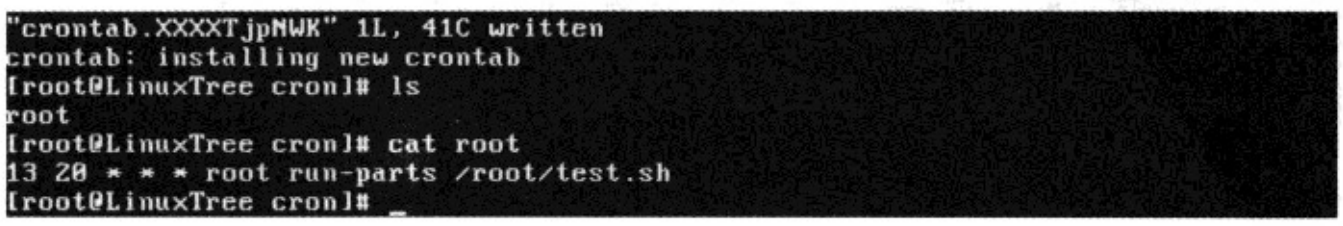


图 7-15：通过 crontab 被产生出的计划任务文件

◆ mail 和 mqueue

这两个目录都是和 email 相关的临时目录，所以放在一起说明，顺便也可以利用 email 来说明【/var/spool】所使用的时候（目录及文件名如图 7-16 所示），日常我们的 email 会经常使用到【/var/spool】，因为当用户 root 寄信给 juergen 时（假设两者的 Mail Server 都为 Linux 操作系统），系统会先将 root 用户所要发送的 mail 先暂存在【/var/spool/mqueue】目录中，等待 Mail Server（像 sendmail）发送出去。

当 Mail Server 将【/var/spool/mqueue】目录中的信件寄达对方（juergen）的信箱时，会发现其实该信箱的地址就是【/var/spool/mail】目录下的 juergen 文件（也就是说，像 juergen@test.com.tw 的意思就是 test.com.tw 主机中的/var/spool/mail/juergen 文件），待 juergen 读取信件时，就会将该文件内容移动至其主目录，或者其 Outlook 目录中，这就是一般 email 大致上的流程。

虽然提了一些 Mail 运行的过程，但这边的重点不只在 Mail 如何进行，也包括 Spool 的作用，也就是如何等待用户将数据取出，另外，如打印机（lpd）也有类似的意思，所以在 Spool 中的数据是要被好好保护的。

```
[root@LinuxTree spool]# ls
anacron  bittorrent  cron  exim  mail   news      repackage  squid
at       clientmqueue cups  lpd   mqueue postfix   samba      vbox
[root@LinuxTree spool]# ls mail/
apache     dovecot    lighttpd  nscd     pound    squid     torrent
avahi      exim       mailnull  ntp      root     sshd      vcsa
backuppc   gdm        mysql     openvpn  rpc      tclhttpd  vdr
boa        haldaemon  named     paul     rpcuser  tcpdump   vdradmin
dbus       hsqldb     newuser   postfix  rpm      test      webalizer
distcache  juergen    nfsnobody postgres smmsp    toncat    xfs
[root@LinuxTree spool]#
```

图 7-16：/var/spool 目录中的内容及 mail 子目录下的文件

7.1.13 /var/tmp

/var/tmp 与/tmp 的主要差异在于，【/var/tmp】是专门为了一些应用程序在安装或执行时，需要在重启后使用某些文件时，能将该文件暂时放置在这目录中，待完成后再行删除；而/tmp 目录则是单纯的临时目录，所有用户皆可使用这一目录，因此，管理员随时会将/tmp 中的文件清空。

因此，两者虽然都放一些临时的文件，但原则上是不同的，只不过是定义上如此，实际上还是根据各软件的设计为主要判断方式，在笔者系统中的【/var/tmp】目录，就只是一个空目录，并无任何临时的文件在其中，如图 7-17 所示。

```
[root@LinuxTree tmp]# find
.
./rpm-tmp.31154
./kdecache-juergen
./kdecache-juergen/ksycocastamp
./kdecache-juergen/ksycoca
[root@LinuxTree tmp]#
```

图 7-17：笔者系统中/var/tmp 下的文件

7.1.14 /var/www

在网页服务器主机中，最重要的就是这一个目录，这里默认存放所有网页服务器的所有网页，可想而知这个目录的重要性。在【/var/www】下并不是直接存放网页，用户必须按照其目录的规则存放到应该存放的目录中（如图 7-18 所示），当然这一切的规则都是可以做配置的，并不是绝对的做法。

```
[root@LinuxTree var]# ls www/
boa  cgi-bin  error  html  icons  manual  mason  tclhttpd  thttpd  usage  wiki
[root@LinuxTree var]#
```

图 7-18：/var/www 目录的文件清单

7.2 挂载用目录【/media vs. /mnt】

在系统使用中一定会碰到许多移动（Removable）储存设备，比如 USB CDROM、USB Key 或 USB Floppy 等，这些设备都是可动态在 Linux 上挂载和卸载（mount 和 umount）的硬件，但每一个设备都需要一个目录去对应，不然用户无法使用，因此，/media 与/mnt 就出现了。

◆ /media

【/media】的出现是为了和/mnt 做区分使用的，除了文件系统外的移动储存设备，现在的操作系统都已经会自动将其挂载到/media 目录下（如图 7-19 所示，不过大部分自动挂载的功能都是在 X Window 下所提供），因为这些设备都属于媒体（Media）设备，一般文件系统类不会在这个目录下出现，所以，/media 的主要服务对象就是刚刚提到的移动储存设备。

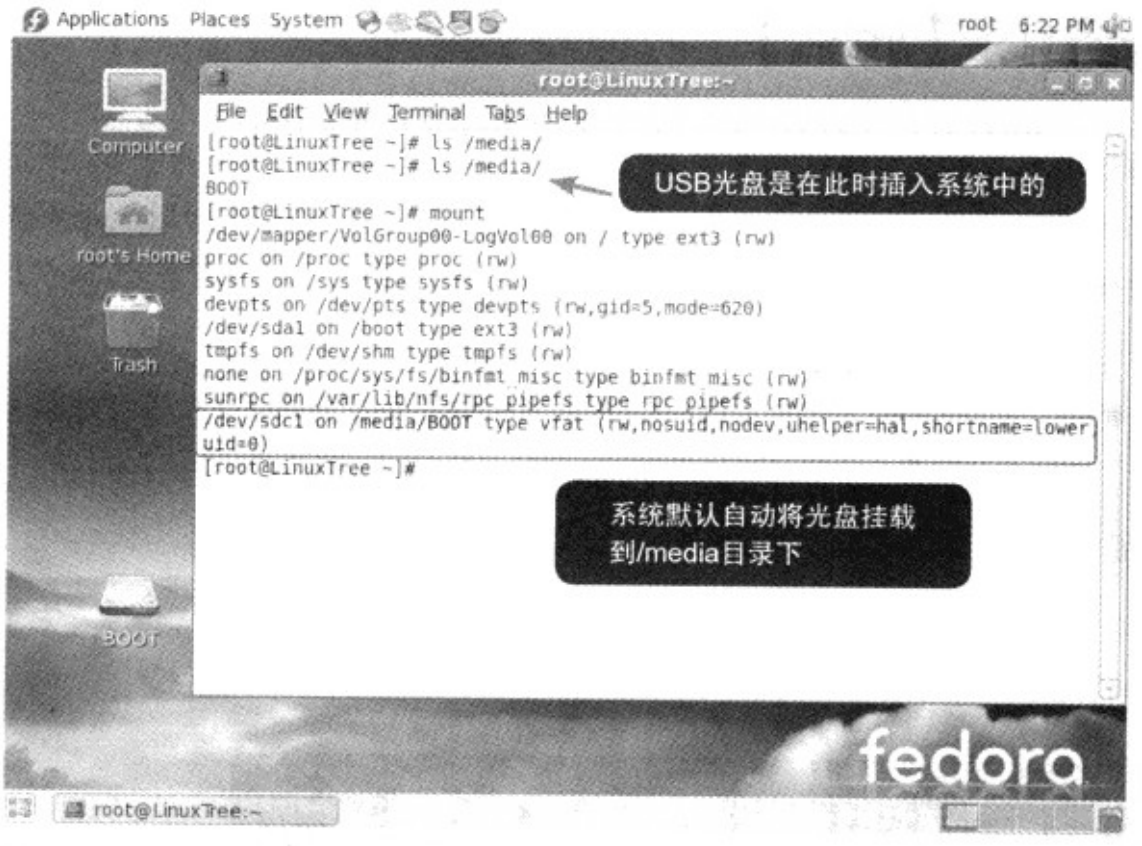


图 7-19：X Window 下的自动挂载功能

◆ /mnt

【/mnt】是早期使用的挂载目录，在其下可以手动或自动建立相关的目录，像 usb、cdrom、floppy 等，但是，目前在/mnt 中所挂载的大多不是外挂的存储设备，很可能是一些 ISO 文件、NFS 文件系统或之类的对象（如图 7-20），因此，/mnt 比较严格的说法，应该说是用来挂载“文件系统”的目录。

/media 与/mnt 的差异其实是很明显的，但因为挂载所使用的目录与用户的习惯息息相关，并且/mnt 是大家行之以久的一个挂载目录，很多时候明明系统已经自动将设备挂载到/media 目录下，但用户会另行再挂载到/mnt 下，只能说这两个目录的使用要等用户习惯后再调整。

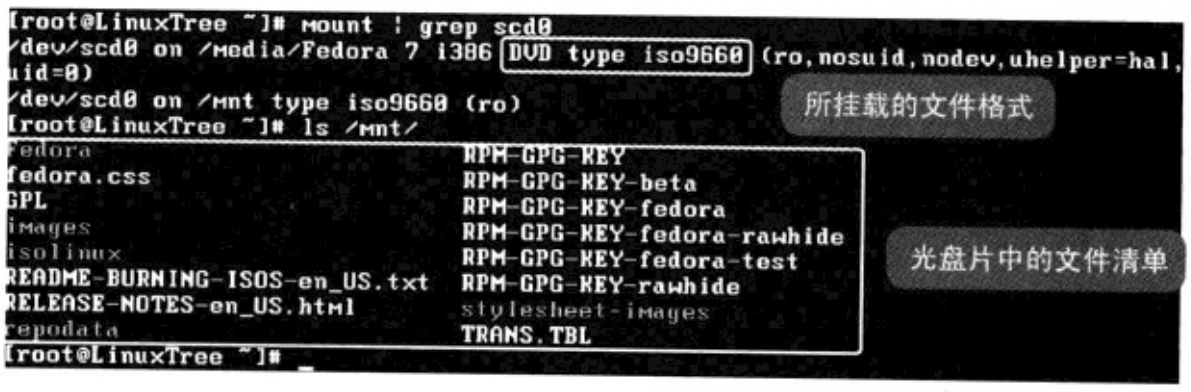


图 7-20：/mnt 的使用时机

7.3 自动挂载服务目录【/misc】

这个目录是一种方便用户的机制，通过这个目录下的子目录，可以让用户随时进入某指定的挂载点，而不须要知道其挂载的格式是什么，另一个重点是当用户尚未进入系统时，是不需要该子目录的，系统会在用户进入的那一瞬间，自动将该子目录建立完成，但前提是【autofs】的服务必须在启动状态。

这听起来可能有点奇怪，但可以看一下图 7-21 中尚未做任何配置时，/misc 目录中与挂载点的状态。

Ⓐ 先确定 autofs 服务是启动中。

Ⓑ /misc 目录中是空的。

Ⓒ 挂载点中没有一个在/misc 目录下。

```
[root@LinuxTree ~]# /etc/rc.d/init.d/autofs status (A)
automount (pid 15315) is running...
[root@LinuxTree ~]# ls /misc/ (B)
[root@LinuxTree ~]# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620) (C)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
[root@LinuxTree ~]#
```

图 7-21：尚未使用 autofs 功能时的状况

这几点确认后，只须记住，autofs 的主要配置文件在【/etc/auto.misc】中（其实有两个，这里只以会修改到的为主），只需要将配置文件稍做修改，就可以让用户轻松使用某些特定的文件系统（如图 7-22 所示）。

Ⓓ 笔者在这个配置文件中加入一行有关“usb”的配置，最前面的“usb”代表在/misc 目录下所要使用的目录名称，中间的“-fstype=ext2”代表该文件系统的格式，最后的“:/dev/sdb1”则代表实际所要指定到的存储设备文件。

Ⓔ 加入后不需要重新启动 autofs 的 daemon，可以实时开始使用，但一开始会发现并没有产生【usb】的目录，因为这时候都还没有任何用户须要使用【usb】所指定的【/dev/sdb1】文件系统。

Ⓕ 当用户试着查看刚刚所配置的【/misc/usb】目录时（这时候并没有这个目录，所以，如果想使用 TAB 键看是没有办法的，必须一个字一个字地打），系统就会自动建立【usb】子目

录，并将【/dev/sdb1】挂载到【/misc/usb】中。

```
[root@LinuxTree ~]# grep ^[^#] /etc/auto.misc
cd              -fstype=iso9660,ro,nosuid,nodev :/dev/cdrom D
usb             -fstype=ext2                :/dev/sdb1
[root@LinuxTree ~]# ls /misc/
[root@LinuxTree ~]# ls /misc/usb E
[root@LinuxTree ~]# ls /misc/
usb                                                        F
[root@LinuxTree ~]# mount
/dev/mapper/VolGroup00-LogVol00 on / type ext3 (rw)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
devpts on /dev/pts type devpts (rw,gid=5,mode=620)
/dev/sda1 on /boot type ext3 (rw)
tmpfs on /dev/shm type tmpfs (rw)
none on /proc/sys/fs/binfmt_misc type binfmt_misc (rw)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
/dev/sdb1 on /misc/usb type ext2 (rw)
[root@LinuxTree ~]#
```

图 7-22：配置完成后的/misc 目录之改变

总结

日志文件在系统管理中是非常实用的一个工具文件，系统管理员可以通过这些文件，对所有的程序、用户、启动过程等详细信息做追踪，将有疑问的点全部抽取出来，这样一来，对系统运行中的状态就可以完全掌握。不过，这些文件当然也需要再做一些细部的权限调整，让用户或侵入者可改变文件的难度再提高，这样才能加强系统的安全性，相对让系统运行的时间延长。

而本章后面所介绍的文件系统或移动储存设备所使用的挂载目录（/media 与/mnt），以及自动挂载的目录“/misc”，都是管理员可以为用户方便性的考量，尤其比如一些机关单位，一般用户（也就是多数为不太会操作 Linux 的用户）众多时，就更需要将这些目录的设计考虑在其中。